ALGORITHMS FOR BALANCED NETWORK FLOW PROBLEM

By

TRINATH NAIK

DEPARTMENT OF INDUSTRIAL AND MANAGEMENT ENGINEERING PROGRAMME

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

APRIL, 1991

ALGORITHMS FOR
BALANCED NETWORK FLOW PROBLEM

A Thesis Submitted

in Partial Fulfillment of the Requirements

for the Degree of

Master of Technology

by

Trinath Naik

to the

Industrial and Management Engineering Programme

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

April, 1991

112183

IMEP-1991-M-NAI-ALG

CERTIFICATE

This is to certify that the work contained in the thesis entitled " **ALGORITHMS FOR BALANCED NETWORK FLOW PROBLEM** ", by **Trinath Naik**, has been carried out under our supervision and that this has not been submitted elsewhere for the award of a degree.

[illegible]

(Dr. R.R.K Sharma)

Lecturer

Industrial and Management

Engineering Programme

Indian Institute Of Technology

Kanpur - 208 016 (U P)

[illegible] 18/4/91

(Dr. R.K Ahuja)

Asst. Professor

Industrial and Management

Engineering Programme

Indian Institute of Technology

Kanpur - 208 016 (U P)

ACKNOWLEDGEMENT

I take this opportunity to express my sincere gratitude and indebtedness to my guide Dr. Ahuja for his valuable suggestions, advices, and constant encouragement during my thesis work. I owe a lot to Dr. Sharma for his timely help and moral support when I needed them most.

I must mention my sincere thanks to all my teachers for making me fit for the thesis work. Finally, I would like to thanks all my friends, particularly, Satya, Sukhram, Sanjay, Rahul for their help and companionship, making my stay here a memorable one.

April, 1991

Trinath Naik

I.I.T Kanpur.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

In this dissertation we have studied the *balanced network flow problem (BNFP)*. In the balanced network flow problem, the objective is to balance the weighted arc flows in a distribution network, by minimizing the difference between the largest and the smallest weighted arc flow. This type of problem arises in military, industrial, and economic applications requiring equitable distribution of resources.

The balanced network flow problem can be transformed to the standard form of the linear programming problem by introducing 2m+2 additional variables and 2m additional constraints, where m is the number of arcs in the network. This transformation is not desirable from computational point of view for large size of m as it increases the size of problem substantially. In this thesis, we present two algorithms, namely, *parametric algorithm* and *binary search algorithm* for the balanced network flow problem, which do not require enlargement of the problem. The parametric algorithm utilizes the relationship between BNFP and *minimax network flow problem (MNFP)* and parametrically solves the latter problem. The binary search algorithm uses binary search technique to reach near the optimal solution. Both the algorithms are coded in FORTRAN-77 and tested extensively with random networks. We compare the computational performance of these algorithms with that of the simplex algorithm, applied to the enlarged linear programming problem. Our computational investigations show that the parametric algorithm is computationally superior than the other two methods.

CHAPTER I

INTRODUCTION

1.1 INTRODUCTION

It is very important to distribute a given amount of resources *equitably* in many situations. By *equitable distribution*, we mean a distribution scheme, which allocates a resource to different centers proportional to their demands. Obviously, equitable distribution arises, whenever there is a scarce resource or nature of the distribution network is such that, it is not possible to satisfy exact requirements of demand centers. Under such conditions, one may like to distribute the resource in such a fashion, that each center get a share proportional to their requirement (equitable distribution). However, the distribution network may not allow the perfect equitableness, when capacity constraints are present. So a distribution scheme might attempt to,

(1) maximize the smallest proportion; or
(2) minimize the largest proportion; or
(3) maximize the smallest proportion and minimize the largest proportion simultaneously.

In this thesis, we discuss a distribution scheme which tries to attain the equitableness by minimizing the difference between the largest and smallest proportions.

1.2 PROBLEM DESCRIPTION

Let $G = (N,A)$ be a directed network with cost c_{ij}, lower bound l_{ij} and capacity u_{ij} associated with each arc $(i,j) \in A$. We associate with each node $i \in N$, an integer number $b(i)$, representing its supply or demand. If $b(i) > 0$, then node i is a *supply node*; if $b(i) < 0$, then it is a *demand node*, and if $b(i) = 0$, then node i is a *transshipment node*. Let $n = |N|$ and $m = |A|$. Let $A' \subseteq A$. Then we define the *Balanced Network Flow Problem (BNFP)* as follows :

$$\text{Minimize} \left\{ \underset{(i,j) \in A'}{\text{Max.}} \left\{ c_{ij}x_{ij} \right\} - \underset{(i,j) \in A'}{\text{Min.}} \left\{ c_{ij}x_{ij} \right\} \right\} \tag{1.1}$$

subject to

$$\sum_{\{j : (i,j) \in A\}} x_{ij} - \sum_{\{j : (j,i) \in A\}} x_{ji} = b(i), \quad \forall\, i \in N, \tag{1.2}$$

$$l_{ij} \le x_{ij} \le u_{ij}, \quad \forall\, (i,j) \in A. \tag{1.3}$$

We refer to the vector $x = (x_{ij})$ as the flow in the network. Each constraint in *(1.2)* implies that the total flow out of a node minus total flow into that node must equal to the net supply / demand of the node. We henceforth refer to these constraints as the *mass balance constraints*. The flow must also satisfy the lower bound and upper bound constraints *(1.3)*, which we refer to as the *capacity constraints*.

1.3 ASSUMPTIONS

Throughout the material in this thesis, the following assumptions have been made.

(1) Although, in general, arc cost c_{ij} can be in the interval $(-\infty, \infty)$ we have considered the common practical case of $c_{ij} \geq 0$ for each $(i,j) \in A$.

(2) To achieve simplifications in the presentation we assume $A = A'$. However we note that this assumption is only for ease of presentation and later we explains how this assumption can be relaxed.

We assume in the rest of the thesis that these assumptions are satisfied unless we state it otherwise.

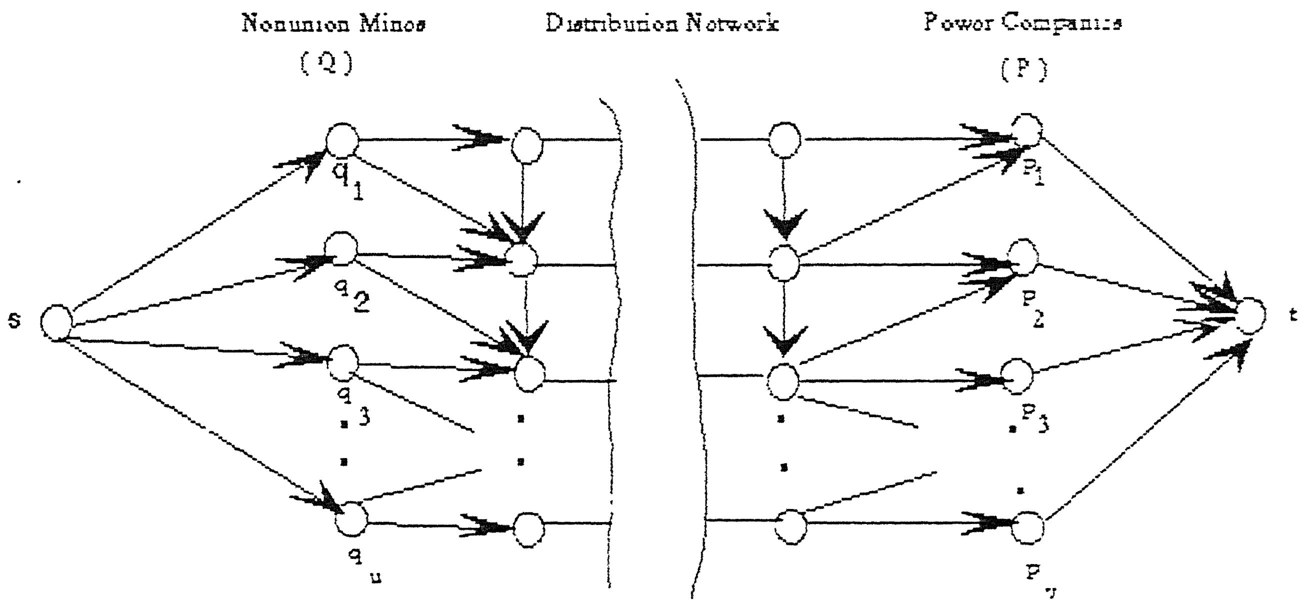
1.4 APPLICATIONS

Resource sharing is a problem arising in numerous application settings. Many important problems such as distributing relief to famine stricken region, allocation of federal government's funds to provinces, distribution of essential commodities in retails, requires sharing of resources. This section illustrates two scenarios where balanced network flow problems may arise in practice.

COAL STRIKE PROBLEM [1]

Consider the situation during a prolonged coal strike.

Some nonunion mines could be producing during the strike. Let Q be the set of nonunion mines and a_q be the amount of coal produced by the mine q for each $q \in Q$. The problem is to distribute the limited coal supply produced by the nonunion mines equitably among the power companies. Distributing each power company the same amount will ignore the size of the company and would allocate smaller companies more coal while the larger one will face shortage. Thus, the distribution scheme must consider the relative size of the power companies. Let P denote the set of the power companies and d_p for $p \in P$ represent the amount of coal used by the power company p during normal times. Let $f(p)$ for $p \in P$ denote the amount of coal the distribution scheme assigns to power company p. So $f(p)/d_p$ is the proportion of the amount of normal time demand supplied to power company p. An equitable distribution scheme would attempt to equalize the proportion. Perfect equalization may not be possible due to capacity constraints of the distribution network. So one can try to balance the proportion by minimizing the difference between the largest and smallest proportion. The problem can be modeled as a BNFP by appropriately defining the network and the terms. Consider a distribution network connecting the nonunion mines and power company as in Figure 1.1. Let s be a super source node and t be a super sink node. Join s to node q by an arc (s,q) for each $q \in Q$. Similarly, join p to node t by an arc (p,t) for each $p \in P$. Now the distribution scheme can be stated in the form of equation (1.1), (1.2), (1.3) where $A' = \{ (p, t) : p \in P \}$, $c_{pt} = 1/d_p$, supply at node s is $b(s) = \sum_{q \in Q} a_q$, demand at t is $b(t) = -b(s)$ and $b(i) = 0$ for other nodes.



Network Model of Coal Strike Example

Figure 1.1

FLOOD RELIEF PROBLEM

Consider a region affected by the flood. There are some relief shelters in that region where the people of the affected areas have taken shelter. Let us denote the shelters as r_1, r_2, ..., r_n. Suppose there are m relief agencies a_1, a_2, ..., a_m operating in these areas. Since each relief agency has limited supply, for effectiveness of their mission they would like each person of all the shelter to get an equal share of the supply. It may not be possible to supply each shelter an amount of supply proportion to the number of people staying in it because of the nature of transportation network. This problem is similar to the coal strike problem if we think of the shelters as the power companies and relief agencies as the nonunion mines.

1.5 MOTIVATION OF THE THESIS

The BNFP can be formulated as a Linear Programming Problem (LP) as below :

$$\text{Minimize } (z - y) \tag{1.4}$$

subject to

$$\sum_{\{j : (i,j) \in A\}} x_{ij} - \sum_{\{j : (j,i) \in A\}} x_{ji} = b(i), \quad \forall\, i \in N, \tag{1.5}$$

$$c_{ij}x_{ij} \leq z, \ \forall\, (i,\ j) \in A, \tag{1.6}$$

$$c_{ij}x_{ij} \geq y, \ \forall\, (i,\ j) \in A, \tag{1.7}$$

$$l_{ij} \leq x_{ij} \leq u_{ij}, \ \forall\, (i,\ j) \in A. \tag{1.8}$$

This linear programming formulation can be converted to the standard form by introducing 2m slack variables. The enlarged LP can be solved by any of the existing simplex based algorithms. These algorithms however consider the constraints (1.6) and (1.7) explicitly which results in the following disadvantages.

(1) As the value of m increases, the size of the basis increases resulting in large storage requirement and more computation per iteration.

(2) The basic solutions are highly degenerate, specially in the first few iterations. So the algorithm may go on iterating without increasing the objective function.

(3) The coefficient matrix of the mass balance constraints possesses special structure. Only 2m out of its nm entries are nonzero. All of its nonzero entries are +1 or -1 and each column has exactly one +1 and one -1. However, the simplex based algorithms for LP does not exploit its special structure to achieve theoretical and computational simplifications.

In this dissertation, we present a simplex based algorithm for the BNFP which takes care of all the disadvantages stated before. It considers the additional constraints (1.6) and (1.7) implicitly and exploits the special structure of the constraint matrix. The algorithm is based on the relationship of BNFP with the Minimax Network Flow Problem (MNFP) and parametrically solving the latter problem.

1.6 NOTATION AND DEFINITIONS

In this section we state without proof some of the basic properties of graphs. The notation and conventions which are used in the thesis are also given for the sake of completeness.

A directed graph $G = (N,A)$ consists of a set, N, of nodes, and a set, A, of arcs whose elements are ordered pairs of distinct nodes. A directed network is a directed graph with numerical values attached to its nodes and/or arcs. As before, we let $n = |N|$ and $m = |M|$. We associate with each arc $(i,j) \in A$ a cost c_{ij}, a capacity u_{ij}, and a lower bound l_{ij}. We assume throughout that $l_{ij} \geq 0$ for each $(i,j) \in A$. Frequently we distinguish two special nodes in a graph : the source s and the sink t.

An arc (i,j) has two end points, i and j. The arc (i,j) is incident to nodes i and j. We refer to node i as the tail node and node j as the head of arc (i,j), and say that the arc (i,j) emanates from node i. The arc (i,j) is an outgoing arc of node i and an incoming arc of node j. The arc adjacency list of node i, $A(i)$, is defined as the set of arcs emanating from node i, i.e., $A(i) = \{(i,j) \in A : j \in N \}$. The degree of a node is the number of incoming and outgoing arc incident to that node.

A directed path $G = (N,A)$ is a sequence of distinct nodes and arcs i_1, (i_1,i_2), i_2, $(i_2,i_3),\ldots,(i_{r-1},i_r)$, i_r satisfying the property that $(i_k,i_{k+1}) \in A$ for each $k = 1,\ldots, r-1$. An undirected path is defined similarly except that for any two consecutive nodes i_k and i_{k+1} on the path, the path contains either arc (i_k,i_{k+1}) or arc (i_{k+1},i_k). We refer to the nodes $i_2, i_3,\ldots,i_{r-1}$ as the internal nodes of the path. A directed cycle is a directed path together with the arc (i_r,i_1) and an undirected cycle is an undirected path together with the arc (i_r,i_1) or (i_1,i_r).

We often use the terminology path to designate either a directed or an undirected path, whichever is appropriate from the context. If any ambiguity arises, we explicitly state directed or undirected path. For the simplicity of notation, we often refer to a path as the sequence of nodes $i_1-i_2-..-i_k$ when its arcs are apparent from the problem context. Alternatively, we sometimes refer to a path as a set of (sequence of) arcs without mention of the nodes. We use similar conventions for representing cycles.

A graph $G' = (N',A')$ is a subgraph of $G = (N,A)$ if $N' \leq A$ and $A' \leq A$. A graph $G' = (N',A')$ is a spanning subgraph of $G = (N,A)$ if $N' = N$ and $A' \leq A$.

Two nodes i and j are said to be connected if the graph contains at least one undirected path from i to j. A graph is said to be connected if all pairs of its nodes are connected : otherwise, it is disconnected. We always assume graph G is connected unless and otherwise stated and hence $m \geq n-1$. We refer to any set $Q \leq A$ with the property that the graph $G' = (N,A-Q)$ is disconnected, and no subset Q has this property, as a cutset of G. A cutset partition the graph into two sets of nodes, X and N-X. We shall alternatively represent the cutset Q as the node partition (X,N-X).

A graph is acyclic if it contains no cycle. A tree is a connected acyclic graph. A subtree of a tree T is a connected subgraph of G. Arcs belonging to T are called nontree arcs. A spanning tree of G = (N,A) has exactly n-1 tree arcs. A node in a tree with degree equal to one is called a leaf node. Each tree has at least two leaf nodes.

A spanning tree contains a unique path between any two nodes. The addition of any nontree arc to a spanning tree creates exactly one cycle. Removing any arc in this cycle again creates a spanning tree. Removing any tree arc from a spanning tree creates two subtrees. Arcs whose end points belong to the two different subtrees constitute a cutset. If any arc belonging to this cutset is added to the subtrees, the resulting graph is again a spanning tree.

1.7 ORGANIZATION OF THE THESIS

A brief chapterwise description of the thesis is given below.

In Chapter 2, the existing literature related to the problem is discussed. The literature is mostly in the area of (1) Maximin linear programming problem, (2) Minimax linear programming problem, (3) Minimax transportation problem, (4) Minimax network flow problem (5) Bottleneck transportation problem, (6) Bottleneck linear programming problem, (7) Balanced linear programming problem.

In Chapter 3, we describe in detail the parametric algorithm and binary search algorithm to solve the balanced network flow problem. The basic ideas and concepts of these algorithms are described, followed by their stepwise descriptions. We illustrate the steps of these algorithms with a numerical example.

In Chapter 4, we give a detail description of our implementations of these algorithms. We also report the computational investigations of these algorithms and the simplex algorithm. The results of our computational testing for different sizes of random networks are also presented.

We give the list of references and bibliography at the end.

CHAPTER II

LITERATURE SURVEY

In this chapter, a brief literature survey in the area of minimax and maximin optimization is presented. Over the years, many researchers have studied the minimax and maximin optimization problems and their results have been reported in the literature. We will go through some of these results in the following discussion.

2.1 MAXIMIN LINEAR PROGRAMMING PROBLEM

Maximin Linear Programming Problem has been studied by Kaplan [2], Brown [3], Gupta and Arora [4], Wu and Posner [5]. A Maximin Linear Programming Problem is described as an optimization problem with a maximin objective function subject to the linear constraints. Mathematically, it can be represented as follows :

$$\text{Maximize } Z = \underset{i:\, i \in (1,\dots,n)}{\text{Minimum}} \{c_i\, x_i\}$$

Subject to

$$A\,X = b,$$

$$X \geq 0,$$

where c_i and x_i are scalars for $i = 1,\dots,n$, X is a column vector of x_i, A is an mxn matrix and b is a column vector of m rows with elements b_i.

Although Maximin Linear Program is mathematically not difficult and is similar to other nonlinear problems that can be solved by linear programming , Kaplan [2] has described a simplified solution procedure when certain constraint conditions exists. He has obtained a simple solution for $c_i \geq 0$ when the following condition holds : There exists a feasible solution X, such that $c_i x_i = t \; \forall \; i = 1, .. ,n$. Such a solution is called a *ray solution* and if it exists, then it clearly solves the Maximin Linear Programming Problem. He has also pointed out many military and economic applications of the problem.

Brown [3] termed the linear programming problem with maximin objective function as "*linear sharing problem*". He has developed an algorithm which alternatively calculates a new global upper bound on the objective function value and then determines if a feasible solution exists that meets the new objective.

Gupta and Arora [4] have extended the work by Kaplan [2] to consider more general cases. Wu and Posner [5] have studied several aspects of maximin linear programming problem. Their algorithm can be divided into two parts. The first part finds an initial feasible solution to maximin linear programming problem. It is essentially an extension of Kaplan's [2] ray solution and whenever such an procedure is not applicable, phase I of the simplex method has been used to find an initial feasible solution of the set $\{ x \mid Ax \leq b \; ; \; x \geq 0 \}$. The second part is an iterative method for searching the optimal solution.

Jocobsen [6], Porteus and Yonmark [7] and Brown [8] have ained minimizations. Ahuja [11] has developed two algorithms, one parametric algorithm which solves linear programming problem with parametric upper bounds and the primal dual algorithm which solves a sequence of related dual feasible linear programming problems. Similar to the minimax linear programming problem, the maximin linear programming problem can be transformed to a linear program by introducing additional constraints. These additional constraints can be viewed as parametric lower bound constraints and can be considered implicitly. In view of this, parametric and primal-dual algorithms of Ahuja [11] can be easily adapted for the maximin linear programming problem.

2.3 MAXIMIN NETWORK FLOW PROBLEM

Minimax / maximin optimization problem in transportation and network context has been studied by various researchers. Brown [12] has considered a flow circulation problem with maximin objective functions. He has partitioned arcs in the network into regular arcs and trade off arcs, where each trade off arc has a nondecreasing trade-off functions associated with it. All arcs have lower and upper bounds on their flows while the value of the smallest trade-off function is maximized. Each cut in the network forms a knapsack sharing problem which leads to an optimality condition similar to maxflow mincut theorem. Brown's flow circulation problem can be stated as :

$$\text{Maximize} \left\{ \underset{(i,j) \in T}{\text{Minimum}}\ \phi_{ij}[x_{ij}] \right\}$$

Subject to

$$\sum_{j \in V} x_{ij} = \sum_{k \in V} x_{ki}, \ \forall\ i \in V$$

$$l_{ij} \le x_{ij} \le u_{ij}, \ \forall\ (i,j) \in R \cup T$$

where $0 \le l_{ij} \le u_{ij}$ for each arc $(i,j) \in R \cup T$ and V is the set of vertices, R is the set of regular arcs and T is the set of trade off arcs, x_{ij} is the flow on arc (i,j), and u_{ij} and l_{ij} represent upper and lower bounds on arc flows.

2.4 MINIMAX NETWORK FLOW PROBLEM

A minimax network flow problem on a network G = (N,A) can be represented mathematically as follows :

$$z^* = \text{Minimize} \left\{ \underset{(i,j) \in A}{\text{maximum}} (c_{ij}\ x_{ij}) \right\}$$

subject to

$$\sum_{(i,j) \in A} x_{ij} = \sum_{(j,i) \in A} x_{ji} \ , j \ne s,t$$

$$\sum_{(s,i) \in A} x_{si} = \sum_{(i,t) \in A} x_{it} = v^*,$$

$$0 \le x_{ij} \le u_{ij} \ , \ (i,j) \in A,$$

where s and t respectively denote the source and sink nodes of the network, x_{ij}, u_{ij}, c_{ij} are flow, capacity and cost (weight) associated with the arc (i,j) respectively.

Stanat and Mago [13] have considered minimax flow in *linear graphs* where a linear graph is defined as a connected acyclic graph with degree of each node either one or two. They have considered a flow problem in which a commodity flows from source nodes to sink nodes. Each source node has a specified value denoting amount of commodity to be disposed of and each sink node has a specified value denoting amount of commodity it can absorb. A solution is optimal, if largest flow along any edge is as small as possible. They have presented an $O(n)$ algorithm for the problem.

Ichimori et al. [14] have developed a polynomial time algorithm for the minimax network flow problem (MNFP). They have called the problem as *weighted minimax flow problem*. The polynomial bound of the algorithm is based on the integral assumption of capacity, cost(weight), and flow value. They have used capacity modification technique which treats capacity as a parameter. Using the concept of maxflow mincut theorem, an interval on z^* is obtained. They have used binary search technique in that interval to obtain z^*. The time complexity of their algorithm is shown to be $O(pn^3)$, where n is the number of nodes in the network. $p = \lceil \log(K) \rceil$, $K = \underset{(i,j)\in A}{\text{Max.}} (c_{ij}u_{ij})$, and $\lceil x \rceil$ means the smallest integer y such that $y \geq x$.

Ichimori et al. [15] have extended their work in [14] to find polynomial algorithm for MNFP with real valued capacity, cost and flow. They have used the capacity modification and strategy for ratio minimization technique of Meggido's [16]. Complexity of this algorithm is shown to be $O(C(n,m)^2)$, where $C(n,m)$ is the the time complexity of maxflow on a graph with n nodes and m arcs. Currently $C(n,m) = O(nm \log(n^2/m))$ [17].

Finke [18] and Tamir [19] have investigated MNFP with integer valued flows. Tamir has demonstrated that bottleneck integral flow can be obtained by solving at most k problems with real valued variables, where k is the number of discrete variables in the problem. Finke [19] has developed a primal algorithm for the integer minimax network flow problem.

2.5 MAXIMUM BALANCED FLOW PROBLEM

Maximum balanced flow problem has been investigated by Minoux [20]. Maximum balanced flow problem is a problem of finding a maximum flow in a two terminal network such that eacn arc flow value is bounded by a fixed proportion of total flow value from source to sink. It can be stated mathematically as follows :

Maximize v

subject to

$$\sum_{j\,:\,(i,j)\in A} x_{ij} - \sum_{j\,:\,(j,i)\in A} x_{ji} = 0, \quad i \in N - \{s,t\},$$

$$\sum_{(s,i)\in A} x_{si} = \sum_{(i,t)\in A} x_{it} = v,$$

$$0 \le x_{ij} \le \alpha_{ij} v,$$

where s and t are source and sink node respectively. α_{ij} is called the balancing rate of the arc (i,j).

Maximum balanced flow problem has been motivated by reliability consideration of communication network. If a balanced flow is assigned in a network it is guaranteed that at most the fixed proportion of flow value is choked when some arc is broken by failure. Fujishige et al. [21] have pointed out the equivalence of the maximum balanced flow problem and MNFP. He has shown that MNFP can be reduced to a variant of the maximum balanced flow problem.

2.6 MINIMAX TRANSPORTATION PROBLEM

Minimax transportation problem (MTP) is a variant of the classical transportation problem and can be stated as follows :

$$\text{Minimize } Z(x) = \underset{(i,j) \in A}{\text{Maximum}} \{c_{ij} \; x_{ij}\}$$

subject to

$$\sum_{j \in J} x_{ij} = a_i, \; \forall \; i \in I,$$

$$\sum_{i \in I} x_{ij} = b_i, \; \forall \; j \in J,$$

$$\sum_{i \in I} a_i = \sum_{j \in J} b_j,$$

where I = { 1,...,m } is a set of origin nodes, J = { m+1,...,m+n }is a set of destination nodes and c_{ij} is the cost of unit shipment on arc (i,j).

Minimax transportation problems have been studied by Goldman [22] and Ahuja [23]. Goldman [22] has considered minimax transportation problem when $c_{ij} = 1 \; \forall \; (i,j) \in A$. Ahuja [23] has developed a parametric algorithm and a primal dual algorithm for MTP. The parametric algorithm solves a transportation problem with parametric upper bounds and primal dual algorithm solves a sequence of maximum flow problem. Worst case complexity of primal-dual algorithm is shown to be $O(n^4)$. He has also described a polynomially bounded algorithm to find an integer optimum solution.

2.7 BOTTLENECK TRANSPORTATION PROBLEM

The bottleneck transportation problem (BTP) can be stated as follows : a set of supplies (a_i) and a set of demands (b_i) are specified such that the total supply is equal to the total demand. There is a transportation time (t_{ij}) associated with each supply point and each demand point. It is required to find a feasible distribution which minimizes the maximum transportation time associated between a supply point and a demand point such that distribution between two points (x_{ij}) is positive, i.e.,

$$\text{Minimize } z = \underset{\{(i,j)\,:\,x_{ij} > 0\}}{\text{Maximum}} \{ t_{ij} \}$$

subject to

$$\sum_{j=1}^{m} x_{ij} = a_i, \quad i = 1,\ldots,m,$$

$$\sum_{i=1}^{m} x_{ij} = b_j, \quad j = 1,\ldots,n,$$

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j ,$$

$$x_{ij} \geq 0 .$$

Many articles on bottleneck transportation problem has been published in literature. A thorough review of literature has been provided by Szwarc [24]. Hammer [25], Szwarc [24] and Srinivasan and Thompson [26] have used primal simplex approaches for solving BTP. They have specialized the primal simplex approaches of linear programming for the transportation problem. The capacitated version of the BTP has been studied by Russel et al.[27].

2.8 BOTTLENECK LINEAR PROGRAMMING PROBLEM

Bottleneck linear programming problem (BLP) has been addressed by Russel et al. [27] and Frieze [28]. The BLP can be stated as

$$\text{Minimize } z = \sum_{i \in \{ j \,:\, c_j = w^* \}} x_i$$

Subject to

$$A x = b,$$

$$x \geq 0,$$

where $w^* = \text{Maximum} \{ c_j, j : x_j \geq 0 \}$, A is an (mxn) matrix, x is an n vector, b is an m vector and c is an n vector. The BLP has wide applications in military operations, transportation of perishable goods [27] and assembly line balancing [28]. Most algorithms of BLP are direct extension of the BTP.

2.9 BALANCED OPTIMIZATION PROBLEMS

Given a finite set E , a vector c of element costs, i.e., $c = \{ c_e : e \in E \}$, and a family F of feasible subsets of E, the balanced optimization problem is to find s^* which minimizes (maximum $\{c_e : e \in S\}$ - minimum $\{c_e : e \in S\}$) over all $s \in F$.

BALANCED ASSIGNMENT PROBLEM

The balanced assignment problem is to find a feasible assignment which minimizes the difference of the values of most expensive and least expensive costs used in the assignment. The balanced assignment problem has been studied by Martello et al. [29]. They have specialized the feasibility questions of the classical assignment problem and solved balanced assignment problem in $O(n^4)$.

BALANCED LINEAR PROGRAMMING PROBLEM

The balanced linear programming problem has been studied by Ahuja [30] and can be stated mathematically as follows :

$$\text{Minimize} \left[\underset{j \in J}{\text{Maximum}}\{c_j \; x_j\} - \underset{j \in J}{\text{Minimum}}\{c_j \; x_j\} \right]$$

subject to

$$Ax = b,$$

$$x \geq 0,$$

where A, b, c, x are $|m\times n|$, $|m\times 1|$, $|1\times n|$, $|n\times 1|$ matrices respectively and $J \subseteq N \equiv \{ 1,2,\ldots,n \}$.

Ahuja [30] has solved BLPP parametrically by transforming the problem into a linear program by introducing additional constraints and treating these constraints as parametric upper and lower bounds. The algorithm is based on a relationship between BLPP and parametric MLPP and parametrically solves the MLPP.

CHAPTER III

ALGORITHMS FOR BALANCED NETWORK FLOW PROBLEM

In this chapter we describe algorithms for the balanced network flow problem (BNFP). We present two algorithms to solve the BNFP, namely *parametric algorithm* and *binary search algorithm*. We discuss the basic ideas and concepts behind the algorithms before their formal statements. Later, we solve a numerical example to illustrate them.

3.1 RAY SOLUTIONS

Although one can always use linear programming approach to solve the BNFP, there are certain cases where due to nature of the constraints, an optimum solution can be found without recourse to any rigorous technique. Kaplan [2] has shown that such a case arises when there exists a feasible *ray solution*, i.e., $c_{ij}x_{ij} = t$ $\forall\ (i,j) \in A$ for some scalar $t \geq 0$. Ray solution, if it exists, clearly solve the BNFP. To identify such a possibility, consider the constraints *(1.2)* and *(1.3)* of the BNFP. Substituting $x_{ij} = t/c_{ij}$ in these constraints, it can be easily shown that an optimum ray solution of BNFP exists if and only if there exists a scalar t which satisfies the following two conditions:

$$t = \frac{b(1)}{\sum\limits_{j \in N} (1/c_{1j}) - \sum\limits_{j \in N} (1/c_{j1})} = \frac{b(2)}{\sum\limits_{j \in N} (1/c_{2j}) - \sum\limits_{j \in N} (1/c_{j2})} = \ldots$$

$$= \frac{b(n)}{\sum\limits_{j \in N} (1/c_{nj}) - \sum\limits_{j \in N} (1/c_{jn})} \qquad (3.1)$$

$$\underset{(i,j) \in A}{\text{Maximum}}(c_{ij} l_{ij}) \le t \le \underset{(i,j) \in A}{\text{Minimum}}(c_{ij} u_{ij}) \qquad (3.2)$$

Obviously conditions *(3.1)* and *(3.2)* are quite unlikel to be satisfied by most BNFP. So rigorous techniques are require to solve them.

It is easy to see that if the BNFP does not possess a ra solution, then any optimum solution is a boundary point of th constraints *(1.2)* and *(1.3)*. For, if (x^*, y^*, z^*) is an optimu solution, which is an interior point, then there exists in th neighborhood of x^* a solution x^o defined as :

$$x^o_{ij} = \begin{cases} x^*_{ij} - \varepsilon/c_{ij} & \text{if } c_{ij}x^*_{ij} = z^* \\ x^*_{ij} + \varepsilon/c_{ij} & \text{if } c_{ij}x^*_{ij} = y^* \\ x^*_{ij} + \alpha_{ij}\varepsilon & \text{otherwise.} \end{cases}$$

Where α_{ij} are such that x^o is feasible to BNFP.

For some arbitrarily small $\varepsilon > 0$, we have $z^o - y^o = z^* - y^* - 2\varepsilon < z^* - y^*$, which contradicts the optimality of x^*.

3.2 PARAMETRIC ALGORITHM

Parametric algorithm of the BNFP is a specialization that for the balanced linear programming problem (BLPP) by Ahu

[30]. The special structure of the network flow problem offers several benefits, particularly streamlining the computations, eliminating the need to explicitly maintain the simplex tableau. The tree structure of the basis permits the parametric algorithm to achieve these efficiencies.

RELATIONSHIP BETWEEN MNFP AND BNFP

It can be easily seen that BNFP, as already stated in Section 1.2, is equivalent to the following linear programming problem.

$$\text{Minimize } (z - y) \tag{3.3}$$

subject to

$$\sum_{j\in N} x_{ij} - \sum_{j\in N} x_{ji} = b(i),\ \forall\ i \in N, \tag{3.4}$$

$$x_{ij} \le \text{Minimum}(d_{ij}z,\ u_{ij}),\ \forall\ (i,j) \in A, \tag{3.5}$$

$$x_{ij} \ge \text{Maximum}(d_{ij}y,\ l_{ij}),\ \forall\ (i,j) \in A, \tag{3.6}$$

where $d_{ij} = 1/c_{ij}\ \forall\ (i,j) \in A$.

The BNFP is closely related to the minimax network flow problem MNFP(y), defined for a parameter y :

$$\text{Minimize } z \tag{3.7}$$

subject to (3.8)

$$\sum_{j\in N} x_{ij} - \sum_{j\in N} x_{ji} = b(i),\ \forall\ i \in N, \tag{3.9}$$

$$x_{ij} \le \text{Minimum}(d_{ij}z,\ u_{ij}),\ \forall\ (i,j) \in A, \tag{3.10}$$

$$x_{ij} \geq \text{Maximum}(d_{ij}y,\ l_{ij}),\ \forall\ (i,j) \in A. \qquad (3.11)$$

Let x(y) is the optimum solution of MNFP(y) with z(y) as the objective function value. The following results establishes the relationship between BNFP and MNFP.

Proposition 3.1. There exists a y such that an optimum solution of MNFP(y) solves BNFP.

Proof: Let (x^*, y^*, z^*) be an optimum solution of BNFP. Now it can be easily seen that (x^*, z^*) is an optimum solution of $MNFP(y^*)$. For if $z(y^*) < z^*$, then the solution $x(y^*)$ is feasible to to BNFP and has objective function value equal to $z(y^*)-y^* < z^*-y^*$, which contradicts the optimality of (x^*, y^*, z^*) for BNFP. ∎

Proposition 3.2. z(y) is a piecewise linear convex function in y.

Proof: It follows from the well known result that the objective function value of linear programming problem (minimization objective) with parametric right hand side is a piecewise linear convex function. For details, see Murty [31]. ∎

Let $0 = e_1 < e_2 < . . . < e_t$ be the values of y where the slope of z(y) changes. Further, let the slope of z(y) in the interval $[e_k, e_{k+1}]$ be u_k for each k = 1, . . . , t-1. Now the optimality criteria for BNFP can be stated by the following theorem.

Theorem 3.1. *If k is the smallest index for which $u_k \geq 1$, then $x(e_k)$ is an optimum solution of BNFP. Further, if $u_{t-1} < 1$, then $x(e_t)$ solves BNFP.*

Proof: As per Proposition 3.1, we can restrict our choice to $x(y)$, $e_0 \leq y \leq e_t$, while looking for an optimum solution of BNFP. In the BNFP, the objective function value of $x(y)$ is $M(y) = z(y)-y$, which is a piecewise linear convex function since it is formed by addition of a piecewise convex function and a linear function. Hence a local optimum solution of $M(y)$ is a global optimum solution. The slope of $M(y)$ in the interval $[e_{k-1}, e_k]$ is $u_{k-1} - 1 < 0$ and in the interval $[e_k, e_{k+1}]$ is $u_k - 1 > 0$. Thus $M(y)$ achieves a global optimum at $y = e_k$ and $x(e_k)$ is an optimum solution of BNFP. Thus the first part of the theorem is proved. The second part of the theorem is obvious in view of the preceding discussion. ∎

PARAMETRIC SOLUTION OF MNFP(Y)

Consider the following network flow problem, called as NFP(y,z), which treats y and z as two parameters.

Minimize $0x$

subject to

$$\sum_{j \in N} x_{ij} - \sum_{j \in N} x_{ji} = b(i), \ \forall \ i \in N, \tag{3.12}$$

$$x_{ij} \leq \text{Minimum}(d_{ij} z, \ u_{ij}), \ \forall \ (i,j) \in A, \tag{3.13}$$

$$x_{ij} \geq \text{Maximum}(d_{ij} y, \ l_{ij}), \ \forall \ (i,j) \in A. \tag{3.14}$$

The above problem is a network flow problem where we are interested in finding a feasible flow satisfying constraints *(3.12)* to *(3.14)*. A basic solution to the above problem is defined by a triple (B,L,U), where B is a spanning tree with n-1 arcs. The subsets B,L, and U partition the set of arcs A into three parts. The set L denotes the set of arcs at their lower bounds and U denotes the set of arcs at their upper bounds. We further divide L into two sets denoted by L_r and L_p, where L_r denotes the set of arcs in L which are at their real lower bounds, i.e., $x_{ij} = l_{ij}$. L_p denotes set of arcs in L which are at their parametric lower bounds, i.e., $x_{ij} = d_{ij}y$. Similarly, we denote U_r and U_p as a set of arcs in U which are at their real upper bounds and parametric upper bounds respectively. Now if we set

$$x_{ij} = u_{ij} \; \forall \; (i,j) \in U_r, \; x_{ij} = d_{ij}z \; \forall \; (i,j) \in U_p$$
$$x_{ij} = l_{ij} \; \forall \; (i,j) \in L_r, \; x_{ij} = d_{ij}y \; \forall \; (i,j) \in L_p$$

then there exists a unique x_{ij} satisfying *(3.12)*. If this x_{ij} further satisfies *(3.13)* and *(3.14)* for all $(i,j) \in B$, then we refer (B,L,U) as an *optimum basis structure of NFP(y,z)*.

<u>*Proposition 3.3.*</u> *(i) If NFP(y,z) has a feasible solution, then there always exists an optimum basis structure (B,L,U).*
(ii) There always exists at least one spanning tree basis B which solves the NFP(y,z).

Proof: We start with x as a feasible solution to NFP(y,z). If the flow in some arc (i,j), x_{ij} = Minimum($d_{ij}z, u_{ij}$), then we put it in U; else if x_{ij} = Maximum($l_{ij}, d_{ij}y$), then we put it in L. The remaining arcs are put in B. Now B may contain a number of cycles. We remove these cycles by selecting one such cycle and sending flow in clockwise (or anti-clockwise) direction in it, until one arc reaches its lower or upper bound. Such an arc is deleted from B and added to either L or U depending upon its bound. We note that by augmenting flow in a cycle we are preserving the mass balance constraint at each node. By repeating the above procedure we get a cycle free solution from an initial feasible solution. Now if the basic arcs in the cycle free solution connect all the nodes, it is clearly a spanning tree solution; otherwise we add some restricted arcs (arcs of L and U) to the basic arcs such that the resulting basis connects all the nodes and it does not contain any cycle. Thus we a get a spanning tree basis and an optimum basis structure (B,L,U) from any feasible solution of NFP(y,z). █

Now in the spanning tree basis, removing any arc from the basis creates two subtrees. Arcs whose nodes belong to two different subtrees form a cut. Thus each basic arc uniquely defines a cut. We represent a cut, formed by removing the basic arc (i,j), as Q_{ij}. We denote N_i and N_j as the set of nodes of the subtrees formed by removing the basic arc (i,j), such that N_i contains node i and N_j contains node j. Clearly, N_i+N_j = N. Let $\overrightarrow{Q}_{ij}$ be a set of arcs in the cut Q_{ij} whose tail nodes lie in N_i and head nodes lie in N_j. Similarly, let $\overleftarrow{Q}_{ij}$ be a set of arcs in the cut set Q_{ij} whose tail nodes lie in N_j and head nodes lie in N_i. Now we can state the following relationship between the above defined terms.

$$Q_{ij} = \left\{(i,j): i \in N_i \text{ and } j \in N_j\right\} + \left\{(i,j): i \in N_j \text{ and } j \in N_i\right\}$$
$$Q_{ij} = \overrightarrow{Q}_{ij} + \overleftarrow{Q}_{ij}$$

Now we establish a result which is crucial for the development of our algorithm.

<u>*Theorem 3.2.*</u> *An optimum basis structure (B,L,U) of NFP(y,z) with the corresponding solution x^* and solution value z^* is an optimum solution of MNFP(y) if and only if there exists a basic arc (p,q) in the basis B satisfying any one of the following two sets of condition:*

(i) $x^*_{pq} = d_{pq}y$ (3.15a)

$(i,j) \in L$ for all $(i,j) \in \overrightarrow{Q}_{pq}$ (3.15b)

$(i,j) \in U$ for all $(i,j) \in \overleftarrow{Q}_{pq}$ (3.15c)

(ii) $x^*_{pq} = d_{pq}z$ (3.16a)

$(i,j) \in L$ for all $(i,j) \in \overleftarrow{Q}_{pq}$ (3.16b)

$(i,j) \in U$ for all $(i,j) \in \overrightarrow{Q}_{pq}$ (3.16c)

Proof: (*Necessity*) We prove it by contradiction. First of all we show that x^* contains at least one parametric upper bound arc. Let x^* be the optimum solution and z^* be the objective function value of MNFP(y), such that it does not have any parametric upper bound arc. Consider the following solution for a small value of ε.

$$x^o = x^* \; ; \; z^o = z^* - \varepsilon \; ;$$

Clearly the solution (x^o, z^o) is feasible to MNFP(y), with objective function value $z^* - \varepsilon$, which contradicts the optimality of (x^*, z^*). Hence, x^* contains at least one parametric upper bound arc.

Now let $d_{ij}y < x^*_{ij} < d_{ij}z \; \forall \; (i,j) \in B$. Add a parametric upper bound arc (g,h) to the basis. Adding (g,h) to the basis will form exactly one cycle. Now augment a flow $\Delta f = d_{gh}\varepsilon$ in this cycle, in a direction opposite to that of (g,h), for a reasonably small value of ε. Repeating the above steps for all the parametric upper bound arcs, we get a solution x^o defined as below:

$$x^o_{ij} = x^*_{ij} \; \forall \; (i,j) \in L \; ; \; x^o_{ij} = x^*_{ij} - \in d_{ij} \; \forall \; (i,j) \in U_p$$
$$x^o_{ij} = x^*_{ij} \; \forall \; (i,j) \in U_r ; \; x^o_{ij} = x^*_{ij} + \varepsilon \, \gamma_{ij} \; \forall \; (i,j) \in B$$

where $\gamma_{ij} = \sum_{(g,h) \in U_p} d_{gh} \, t_{ij,gh}$;

$t_{ij,gh} = 0$ if the basic arc (i,j) is not a member of the cycle formed by adding nonbasic arc (g,h) to the basis;

$t_{ij,gh} = 1$ if (i,j) is a member of the cycle and the orientation of (i,j) is opposite to that of (g,h);

$t_{ij,gh} = -1$ if (i,j) is a member of the cycle and the orientation of (i,j) is same as that of (g,h).

It is clear that the solution x^o is feasible to the MNFP(y) and has the objective function value z-ε, which contradicts the optimality of the solution x^*. Hence there must exist at least one basic arc, say (p,q), for which $x_{pq} = d_{pq}y$ or $x_{pq} = d_{pq}z$.

Let there be exactly one basic arc for which $x_{pq} = d_{pq}y$. Further, let condition (3.15b) be violated for some arc $(k,l) \in \overrightarrow{Q}_{pq}$. This means that $(k,l) \in U$. Using the argument as before we obtain the solution x^o. Consider the flow value of basic arc (p,q) in the solution x^o:

$$x^o_{pq} = d_{pq}y + \varepsilon\,\gamma_{pq} \quad \text{where } \gamma_{pq} = \sum_{(g,h)\in U_p} d_{gh}\, t_{pq,gh}$$

If $\gamma_{pq} \geq 0$, then x^o is feasible to the MNFP(y) and has the objective function value z-ε. When $\gamma_{pq} \leq 0$, add the arc (k,l) to the basis, this forms a cycle. Now augmenting a flow equals to $-\varepsilon\gamma_{pq}$ in this cycle in the direction of (p,q) we obtain a solution $\hat{x}$ which is defined as below:

$$\hat{x}_{ij} = x^*_{ij},\ \forall\ (i,j) \in (L \cup U)-\{(k,l)\};$$
$$\hat{x}_{kl} = x^*_{kl} + \gamma_{pq}\varepsilon;$$
$$\hat{x}_{ij} = x^*_{ij} + \gamma_{ij}\,\varepsilon + t_{ij,pq}\,\gamma_{pq}\varepsilon,\ \forall\ (i,j) \in B - \{(p,q)\}$$
$$\hat{x}_{pq} = x^*_{pq} = d_{pq}y$$

It can be easily verified that $\hat{x}$ is feasible to the MNFP(y) and has the objective function value $z - \varepsilon$, which again contradicts the optimality of solution x^*. A similar reasoning can be used to show the necessity of the remaining cases. ∎

(Sufficiency) Now we will show that if there exists a basic arc, say (p,q), in the basis which satisfies condition *(3.15)* or *(3.16)*, then the corresponding solution x^* and solution value z^* solves the MNFP(y).

Let a basic arc (p,q) satisfies condition *(3.15)* and the corresponding solution is x^*. Adding the mass balance constraints *(3.4)* for all the nodes in N_q we get

$$\sum_{i \in N_q} b(i) = \sum_{(i,j)\in \overleftarrow{Q}_{pq}} x^*_{ij} - x^*_{pq} - \sum_{(i,j)\in(\overrightarrow{Q}_{pq} - \{(p,q)\})} x_{ij} \qquad (3.17)$$

Substituting $x^*_{pq} = d_{pq}y$ in the above equation and rearranging both the both sides we obtain:

$$d_{pq}y = \sum_{i \in N_q} b(i) - \sum_{(i,j)\in \overleftarrow{Q}_{pq}} x^*_{ij} + \sum_{(i,j)\in(\overrightarrow{Q}_{pq} - \{(p,q)\})} x^*_{ij} \qquad (3.18)$$

Let x^o be an optimal solution of the MNFP(y). Now we can state the following relationship between x^o and x^* :

1. $x^o_{ij} = x^*_{ij} + \Delta_{ij}$, $\forall\ (i,j) \in L$, since x^o is feasible tc the MNFP(y)
2. $x^o_{ij} = x^*_{ij} - \Delta_{ij}$, $\forall\ (i,j) \in U$, since x^* is an upper bound

on the optimum solution of the MNFP(y)

where $\Delta_{ij} \geq 0 \ \forall \ (i,j) \in L \cup U$. Similarly, adding the mass balance constraints *(3.4)* of all nodes in N_q for the solution x^o we get:

$$x^o_{pq} = \sum_{i \in N_q} b(i) - \sum_{(i,j) \in \overleftarrow{Q}_{pq}} (x^*_{ij} + \Delta_{ij}) + \sum_{(i,j) \in (\overrightarrow{Q}_{pq} - \{(p,q)\})} (x^*_{ij} - \Delta_{ij}) \qquad (3.19)$$

On simplification, we get:

$$x^o_{pq} = d_{pq} y - \sum_{(i,j) \in (Q_{pq} - \{(p,q)\})} \Delta_{ij} \qquad (3.20)$$

It can be seen that any positive value of Δ_{ij} makes the solution x^o infeasible to the MNFP(y). Hence $\Delta_{ij} = 0 \ \forall \ (i,j) \in L \cup U$. So the solution x^o has same objective function value as that of x^*. A similar proof can be given when condition *(3.16)* is satisfied. ∎

We call an optimum basis structure of the NFP(y,z) satisfying optimality condition of Theorem 3.2 as *optimum basis structure of MNFP(y)*. We call the basic arc (p,q) satisfying the optimality condition as *critical arc* and the corresponding cut Q_{pq} as *critical cut*. It may be noted that the critical cut satisfies the following property : (a) All the arcs in the critical cut are restricted, i.e., at their upper bounds or lower bounds. (b) All the upper bound arcs are in the same direction, all the lower bound arcs are also in the same direction and it is opposite to the orientation of all upper bound arcs.

Let (B,L,U) denote the optimum basis structure of the MNFP(y) at y = $\underline{y}$ with Q_{pq} as the critical cut. Let $\underline{x}$ be the corresponding solution and $\underline{z}$ be the objective function value. Please note that this basis structure can be obtained by solving a MNFP as described in Chapter 4. We now determine the maximum value of y, denoted by $\overline{y}$, such that (B,L,U) remains an optimum basis structure of the MNFP(y) for every $y \in (\underline{y}, \overline{y})$. The interval $(\underline{y}, \overline{y})$ is called the *characteristic interval* of (B,L,U).

DETERMINING THE CHARACTERISTIC INTERVAL

The characteristic interval of (B,L,U) is determined from the following considerations:

(1) Increase y by an amount Δy, set $x_{ij} = d_{ij}(y + \Delta y)$ for each $(i,j) \in L_p$.

(2) The value of z increases by an amount, say Δz, and flows on all parametric upper bound arcs are set at their changing upper bound, i.e., $x_{ij} = d_{ij}(z+\Delta z)$, $\forall\ (i,j) \in U_p$.

(4) The flow value on each basic arc, say (i,j), changes uniquely as the y value increases by one unit. Let us denote this changes by s_{ij}.

(5) The relationship between Δz and Δy is determined from the criteria that Q_{pq} remains the critical cut. Let us denote the ratio $\Delta z/\Delta y$ by θ.

(6) The maximum permissible value of Δy is determined from the fact that the flow value on each basic arc remains within its permissible bounds.

COMPUTING THE RATIO $\Delta Z/\Delta Y$

Consider the arcs in the critical cut. As y increases by an amount Δy, all the parametric lower bound arcs in the critical cut carry an additional amount of $d_{ij}\Delta y$ from one subtree to another subtree. Similarly, the parametric upper bound arcs in the critical cut carry an additional amount of flow $d_{ij}\Delta z$ each in the reverse direction. In order to satisfy the mass balance constraints *(3.4)* these additional incoming and outgoing flows should balance each other. This criteria is used to compute $\Delta z/\Delta y$. Thus $\Delta z/\Delta y$ can be computed depending upon the following cases:

Case I. When the critical arc (p,q) is at its parametric lower bound.

Equating the additional incoming and outgoing flow we get

$$\sum_{(i,j)\in \overrightarrow{Q}_{pq}\cap L_p} d_{ij}\Delta y + d_{pq}\Delta y = \sum_{(i,j)\in \overleftarrow{Q}_{pq}\cap U_p} d_{ij}\,\Delta z$$

$$\Rightarrow \Delta z/\Delta y = \frac{\sum_{(i,j)\in \overrightarrow{Q}_{pq}\cap L_p} d_{ij} + d_{pq}}{\sum_{(i,j)\in \overleftarrow{Q}_{pq}\cap U_p} d_{ij}} \qquad (3.21)$$

Case II. When the critical arc (p,q) is at its parametric upper bound.

$$\Delta z/\Delta y = \frac{\sum\limits_{(i,j)\in \overleftarrow{Q}_{pq}\cap L_p} d_{ij}}{\sum\limits_{(i,j)\in \overrightarrow{Q}_{pq}\cap U_p} d_{ij} + d_{pq}} \tag{3.22}$$

It may be noted that if $\theta \geq 1$, then as per Theorem 3.1, the corresponding solution $\underline{x}$ is optimum to BNFP and the algorithm can be terminated at this stage; otherwise it should be continued.

Increasing y by an amount Δy causes an increase in flow on parametric lower bound arcs by $d_{ij}\Delta y$ and that on parametric upper bound arcs by $d_{ij}\Delta z$. This increase in flow on parametric arcs causes excess or deficit at a node, which is to be balanced by corresponding flow changes in basic arcs. So as y increases by one unit, flow in each basic arc changes in a unique way. It is not difficult to see that as long as the current basis structure remains feasible, the flow on each basic $(i,j) \in B$ is of the form $\underline{x}_{ij} + s_{ij}\,\Delta y$.

COMPUTING FLOW CHANGES IN BASIC ARCS (S_{IJ})

In order to determine the flow changes on a basic arc (i,j), as y increases by one unit, following steps can be used.

Step I. Set $\Pi_i = 0 \ \forall\ i \in N$ and perform the following operations:

(a) For each $(i,j) \in L_p$:

$\Pi_i = \Pi_i - d_{ij}$,

$\Pi_j = \Pi_j + d_{ij}$.

(b) For each $(i,j) \in U_p$

$\Pi_i = \Pi_i - \theta\, d_{ij}$,

$\Pi_j = \Pi_j + \theta\, d_{ij}$.

Step II. In the spanning tree corresponding to the basis select a node of degree one, say i, and find a unique basic arc (g,h) such that $g = i$ or $h = i$. If $g = i$, then set $s_{gh} = \Pi_i$; else $s_{gh} = -\Pi_i$. Modify $\Pi_g = \Pi_g + \Pi_h$. Delete the arc (g,h) from the basis and repeat this step, until no arc remains in the basis.

At $y = \bar{y}$, flow on a basic arc, say (i,j), reaches either its lower or upper bound. The arc (i,j) is then dropped from the basis and an eligible nonbasic arc is selected to enter the basis.

FINDING LEAVING ARC IN THE BASIS

A basic arc leaves the basis if it obstructs the increase in y value. A leaving arc (i,j) is determined from the criteria that the flow on a basic arc lies within its upper and lower bound, i.e.,

$$\text{Maximum}(l_{ij}, d_{ij}(\underline{y}+\Delta y)) \le \underline{x}_{ij} + s_{ij}\Delta y \le \text{Minimum}(d_{ij}(\underline{z}+\Delta z), u_{ij}), \ \forall \ (i,j) \in B. \quad (3.23)$$

Using the above four inequalities we obtain:

$$\Delta y^1_{ij} = \begin{cases} \dfrac{\underline{x}_{ij} - d_{ij}\,\underline{y}}{(-s_{ij} + d_{ij})} & \text{if } (-s_{ij}+d_{ij}) > 0 \\ \infty & \text{otherwise} \end{cases}$$

$$\Delta y^2_{ij} = \begin{cases} \dfrac{\underline{x}_{ij} - d_{ij}\,\underline{z}}{(-s_{ij} + \theta\, d_{ij})} & \text{if } (-s_{ij}+\theta\, d_{ij}) < 0 \\ \infty & \text{otherwise} \end{cases}$$

$$\Delta y^3_{ij} = \begin{cases} \dfrac{l_{ij} - \underline{x}_{ij}}{s_{ij}} & \text{if } s_{ij} < 0 \\ \infty & \text{otherwise} \end{cases}$$

$$\Delta y^4_{ij} = \begin{cases} \dfrac{u_{ij} - \underline{x}_{ij}}{s_{ij}} & \text{if } s_{ij} > 0 \\ \infty & \text{otherwise} \end{cases}$$

Let $\Delta y_{ij} = \text{Minimum}\{ \Delta y^1_{ij}, \Delta y^2_{ij}, \Delta y^3_{ij}, \Delta y^4_{ij} \}$. Further, let $\Delta y^b = \text{Minimum}\{\Delta y_{ij}\}, \ \forall \ (i,j) \in B$. While determining the permissible increase in $\underline{y}$, i.e., Δy, we have to consider nonbasic arcs as well. The Δy should be such that a nonbasic arc does not change from its parametric bound to its real bound or vice versa, for whenever such a change occurs, the s_{ij} values of basic arcs change. There are two such possibilities:

(1) A real lower bound arc reaches its parametric lower bound.

(2) A parametric upper bound arc reaches its real upper bound.

Considering these possibilities we obtain:

(1) $\Delta y^{l}_{ij} = c_{ij} l_{ij} - y, \ \forall\ (i,j) \in L - L_p$;

(2) $\Delta y^{u}_{ij} = (c_{ij} u_{ij} - z)/\theta \ \ \forall\ (i,j) \in U_p$.

Let $\Delta y^{l} = \text{Minimum}(\Delta y^{l}_{ij}), \ \forall\ (i,j) \in L - L_p$;

$\Delta y^{u} = \text{Minimum}(\Delta y^{u}_{ij}), \ \forall\ (i,j) \in U_p$;

$\Delta y = \text{Minimum}(\Delta y^{l}, \Delta y^{u}, \Delta y^{b})$.

Now $\bar{y} = \underline{y} + \Delta y$. Clearly, (B,L,U) is an optimum basis structure of MNFP(y) for all $y \in (\underline{y}, \bar{y})$ with optimum objective function value $\underline{z} + \theta\ (y - \underline{y})$. The $\bar{z} = \underline{z} + \theta\ (\bar{y} - \underline{y})$ is the objective function value and $\bar{x}$ is the optimum solution at $y = \bar{y}$. Let (k,l) be the arc for which minimum Δy is achieved. There are three possible cases:

(1) $\Delta y = \Delta y^{l} = \Delta y^{l}_{kl}$ <=> A real lower bound arc (k,l) reaches its parametric lower bound.

(2) $\Delta y = \Delta y^{u} = \Delta y^{u}_{kl}$ <=> A parametric upper bound arc (k,l) reaches its real upper bound.

(3) $\Delta y = \Delta y^{b} = \Delta y^{b}_{kl}$ <=> A basic arc (k,l) reaches its lower or upper bound.

If case *1* or case *2* occurs, then the following steps can be followed. If $(k,l) \in Q_{pq}$, i.e., an nonbasic arc in the critical cut changes its bound, then fresh computation of the θ and s_{ij} is required; otherwise the θ does not changes and only updating the s_{ij} values are required. We describe below a procedure to update the s_{ij} values efficiently.

```
Procedure    UPDATE ;
    If ($\Delta y = \Delta y^l$) then
        Let (i,j) be the arc for which $\Delta y^l_{ij} = \Delta y^l$
        Entering arc (i,j) to the basis B will form a cycle
        For all basic arc (g,h) in that cycle
            If orientation of (i,j) and (g,h) are same then
                $s_{gh} = s_{gh} + d_{ij}$
            Else
                $s_{gh} = s_{gh} - d_{ij}$
            Endif
    Else
        Let (i,j) be the arc for which $\Delta y^u_{ij} = \Delta y^u$
        Entering arc (i,j) to the basis B will form a cycle
        For all basic arc (g,h) in that cycle
            If orientation of (i,j) and (g,h) are same then
                $s_{gh} = s_{gh} - d_{ij}\theta$
            Else
                $s_{gh} = s_{gh} + d_{ij}\theta$
            Endif
    Endif
```

If case 3 occurs, then basic arc (k,l) leaves the basis depending upon the following conditions.

1. If $\Delta y_{kl} = \Delta y_{kl}^{1}$, then arc (k,l) leaves the basis at its parametric lower bound.
2. If $\Delta y_{kl} = \Delta y_{kl}^{2}$, then arc (k,l) leaves the basis at its parametric upper bound.
3. If $\Delta y_{kl} = \Delta y_{kl}^{3}$, then arc (k,l) leaves the basis at its real lower bound.
4. If $\Delta y_{kl} = \Delta y_{kl}^{4}$, then arc (k,l) leaves the basis at its real upper bound.

If the leaving arc (k,l) is the critical arc (p,q) itself, it implies that $\bar{x}_{kl} = d_{kl}\bar{y} = d_{kl}\bar{z} \Leftrightarrow \bar{y} = \bar{z}$. Thus the current solution is a *ray solution* and the algorithm can be terminated.

FINDING ENTERING ARC TO THE BASIS

When the basic arc (k,l) leaves the basis, a cut Q_{kl} is produced. Now a nonbasic arc is selected to enter the basis such that the spanning tree property of the basis is maintained and the entering arc does not restrict the increase in y value. An entering arc is selected depending upon the following cases.

Case I. When basic arc (k,l) leaves the basis at its lower bound

When the basic arc (k,l) leaves the basis at its lower bound, a nonbasic arc becomes eligible to enter the basis if it is a member of cut Q_{kl}, for if any other arc enters the basis, the spanning tree property of the basis would not be maintained. Two types of arcs in the cut Q_{kl} are eligible to enter the basis, they are, (1) a lower bound arc whose orientation is opposite to that of the arc (k,l), (2) a upper bound arc whose orientation is same as that of the arc (k,l). It may be noted that if any other arc enters the basis it will restrict further increase in y value. So we define a set of eligible entering arcs K as below:

$$K = \left\{ (i,j) : (i,j) \in L \cap \overleftarrow{Q}_{kl} \text{ or } (i,j) \in U \cap \overrightarrow{Q}_{kl} \right\}$$

Case II. When basic arc (k,l) leaves the basis at its upper bound

Similarly, we define the eligible entering arc set K as below:

$$K = \left\{ (i,j) : (i,j) \in L \cap \overrightarrow{Q}_{kl} \text{ or } (i,j) \in U \cap \overleftarrow{Q}_{kl} \right\}$$

While selecting an entering arc from the set K, preference is given to the arcs which are not members of the critical cut. This ensures that the critical cut continues to satisfy the optimality condition after a basis change. However, when all the arcs in K are member of the critical cut, i.e., $K \subseteq Q_{pq}$, then any arc from K is selected; in this case, though the critical arc remain the same, the critical cut changes. When K is empty, Q_{kl}

satisfies the optimality condition, and it is made the new critical cut and corresponding arc is made new critical arc.

ALGORITHM STATEMENT

We summarize the preceding discussion by the following stepwise description of the algorithm.

Step 1. Set $\underline{y}$ = Minimum(c_{ij}/l_{ij}) $\forall$ $(i,j) \in A$. Solve MNFP with $y = \underline{y}$. If the problem is infeasible, go to step 6; else let $\underline{x}$ and $\underline{z}$ be the solution and the objective function value respectively. Obtain an optimum basis structure (B,L,U) with arc (p,q) as the critical arc and Q_{pq} as the critical cut.

Step 2. Compute the ratio $\theta = \Delta z/\Delta y$ depending upon the following cases:

Case 1. Critical arc is at its parametric lower bound, i.e., $x_{pq} = d_{pq}y$

$$\theta = \frac{\sum\limits_{(i,j) \in \overrightarrow{Q}_{pq} \cap L_p} d_{ij} + d_{pq}}{\sum\limits_{(i,j) \in \overleftarrow{Q}_{pq} \cap U_p} d_{ij}}$$

case 2. *Critical arc is at its parametric upper bound, i.e.,* $x_{pq} = d_{pq}z$

$$\theta = \frac{\sum_{(i,j) \in \overleftarrow{Q}_{pq} \cap L_p} d_{ij}}{\sum_{(i,j) \in \overrightarrow{Q}_{pq} \cap U_p} d_{ij} + d_{pq}}$$

Step 3. *Compute* S_{ij} *(Flow change in basic arc (i,j) as y increases by one unit) for each basic arc in the following manner:*

(a) Set $\Pi_i = 0 \ \forall \ i \in N$ and compute the following:
$\Pi_i = \Pi_i - d_{ij}$ and $\Pi_j = \Pi_j + d_{ij}$; $\forall \ (i,j) \in L_p$
$\Pi_i = \Pi_i - \theta \ d_{ij}$ and $\Pi_j = \Pi_j + \theta \ d_{ij}$; $\forall \ (i,j) \in U_p$

(b) In the spanning tree corresponding to the basis select a node of degree one, say i, and find the unique basic arc (g,h) such that $g = i$ or $h = i$. If $g = i$, then set $s_{gh} = \Pi_i$; else $s_{gh} = -\Pi_i$. Modify $\Pi_g = \Pi_g + \Pi_h$. Delete basic arc (g,h) from the basis and repeat this step, until no arc remains in the basis.

Step 4 *Finding the leaving arc*

For each $(i,j) \in B$, compute the following:

$$\Delta y^1_{ij} = \begin{cases} \dfrac{\underline{x}_{ij} - d_{ij}y}{(-s_{ij} + d_{ij})} & \text{if } (-s_{ij} + d_{ij}) > 0 \\ \infty & \text{otherwise} \end{cases}$$

$$\Delta y^2_{ij} = \begin{cases} \dfrac{\underline{x}_{ij} - d_{ij}z}{(-s_{ij} + \theta\, d_{ij})} & \text{if } (-s_{ij} + \theta\, d_{ij}) < 0 \\ \infty & \text{otherwise} \end{cases}$$

$$\Delta y^3_{ij} = \begin{cases} \dfrac{l_{ij} - \underline{x}_{ij}}{s_{ij}} & \text{if } s_{ij} < 0 \\ \infty & \text{otherwise} \end{cases}$$

$$\Delta y^4_{ij} = \begin{cases} \dfrac{u_{ij} - \underline{x}_{ij}}{s_{ij}} & \text{if } s_{ij} > 0 \\ \infty & \text{otherwise} \end{cases}$$

Let $\Delta y_{ij} = \text{Minimum } (\Delta y^1_{ij}, \Delta y^2_{ij}, \Delta y^3_{ij}, \Delta y^4_{ij})$;

$$\Delta y^b = \underset{(i,j) \in B}{\text{Minimum}} \; (\Delta y_{ij}).$$

For each $(i,j) \in L - L_p$, compute $\Delta y^l_{ij} = c_{ij} l_{ij} - y$.

Let $\Delta y^l = \text{Minimum}(\Delta y^l_{ij})$, $\forall\, (i,j) \in L - L_p$.

For each $(i,j) \in U_p$ compute $\Delta y^u_{ij} = \dfrac{(c_{ij} u_{ij} - z)}{\theta}$.

Let $\Delta y^u = \text{Minimum}(\Delta y^u_{ij})\ \forall\, (i,j) \in U_p$.

Let $\Delta y = \text{Minimum}(\Delta y^b, \Delta y^l, \Delta y^u)$. Set $\underline{y} = \underline{y} + \Delta y$, $\underline{z} = \underline{z} + \theta\, \Delta y$, $\underline{x}_{ij} = \underline{x}_{ij} + s_{ij}\, \Delta y\ \forall\, (i,j) \in B$. Let (k,l) be the arc for which minimum Δy is achieved.

```
If $\Delta y = \Delta y^l$ or $\Delta y = \Delta y^u$, then
    If $\Delta y = \Delta y^l$, lower bound arc $(k,l)$ reaches its
    parametric lower bound.
    If $\Delta y = \Delta y^u$, then parametric upper bound arc $(k,l)$
    reaches its real upper bound.
    If $(k,l) \in Q_{pq}$, then
        go to step 2
    Else use UPDATE to modify $s_{ij}$ values and
        go to the beginning of step 4.
    Endif
Else basic arc $(k,l)$ reaches its bound depending on the
    following conditions.
    If $\Delta y_{kl} = \Delta y^1_{kl}$, then $(k,l)$ leaves the basis at
    parametric lower bound.
    If $\Delta y_{kl} = \Delta y^2_{kl}$, then $(k,l)$ leaves the basis at its
    parametric upper bound.
    If $\Delta y_{kl} = \Delta y^3_{kl}$, then $(k,l)$ leaves the basis at its
    real lower bound.
    If $\Delta y_{kl} = \Delta y^4_{kl}$, then $(k,l)$ leaves the basis at its
    real upper bound.
Endif
```

Step 5. If the leaving arc (k,l) is the critical arc (p,q) itself, the current solution is a ray solution, go to step 7; else define an eligible entering arc set K as follows:

$$K = \begin{bmatrix} (i,j) : (i,j) \in L \cap \overleftarrow{Q}_{kl} \text{ or } (i,j) \in U \cap \overrightarrow{Q}_{kl} \\ \text{if } \Delta y_{kl} = \Delta y^1_{kl} \text{ or } \Delta y^3_{kl} \\ (i,j) : (i,j) \in L \cap \overrightarrow{Q}_{kl} \text{ or } (i,j) \in U \cap \overleftarrow{Q}_{kl} \\ \text{if } \Delta y_{kl} = \Delta y^2_{kl} \text{ or } \Delta y^4_{kl} \end{bmatrix}$$

If K is empty, make Q_{kl} the new critical cut, go to step 2; else find an entering arc from K. While selecting an entering arc, preference is given to the arcs which are not members of the critical cut. If all elements of K are members of the critical cut, i.e., $K \subseteq Q_{pq}$, then select any arc from K to enter the basis. Find the new critical cut and go to step 2, else go to step 3 for fresh computations of s_{ij}.

Step 6. The BNFP is infeasible, STOP.

Step 7. *(Optimal solution)* The optimal solution is $\underline{x}_{ij} = d_{ij}\underline{y}$ $\forall$ $(i,j) \in L_p$, $\underline{x}_{ij} = d_{ij}\underline{z}$ $\forall$ $(i,j) \in U_p$, $\underline{x}_{ij} = l_{ij}$ $\forall$ $(i,j) \in L - L_p$, $\underline{x}_{ij} = u_{ij}$ $\forall$ $(i,j) \in U - U_p$ and $\underline{x}_{ij} = \underline{x}_{ij}$ $\forall$ $(i,j) \in B$. The objective function value is $\underline{z} - \underline{y}$.

The preceding steps of the parametric algorithm is represented by the flow chart shown in Figure 3.1.

It may be recalled, from Section 1.3, our assumptions that BNFP is defined over all the arcs in a network, i.e., $A' = A$.

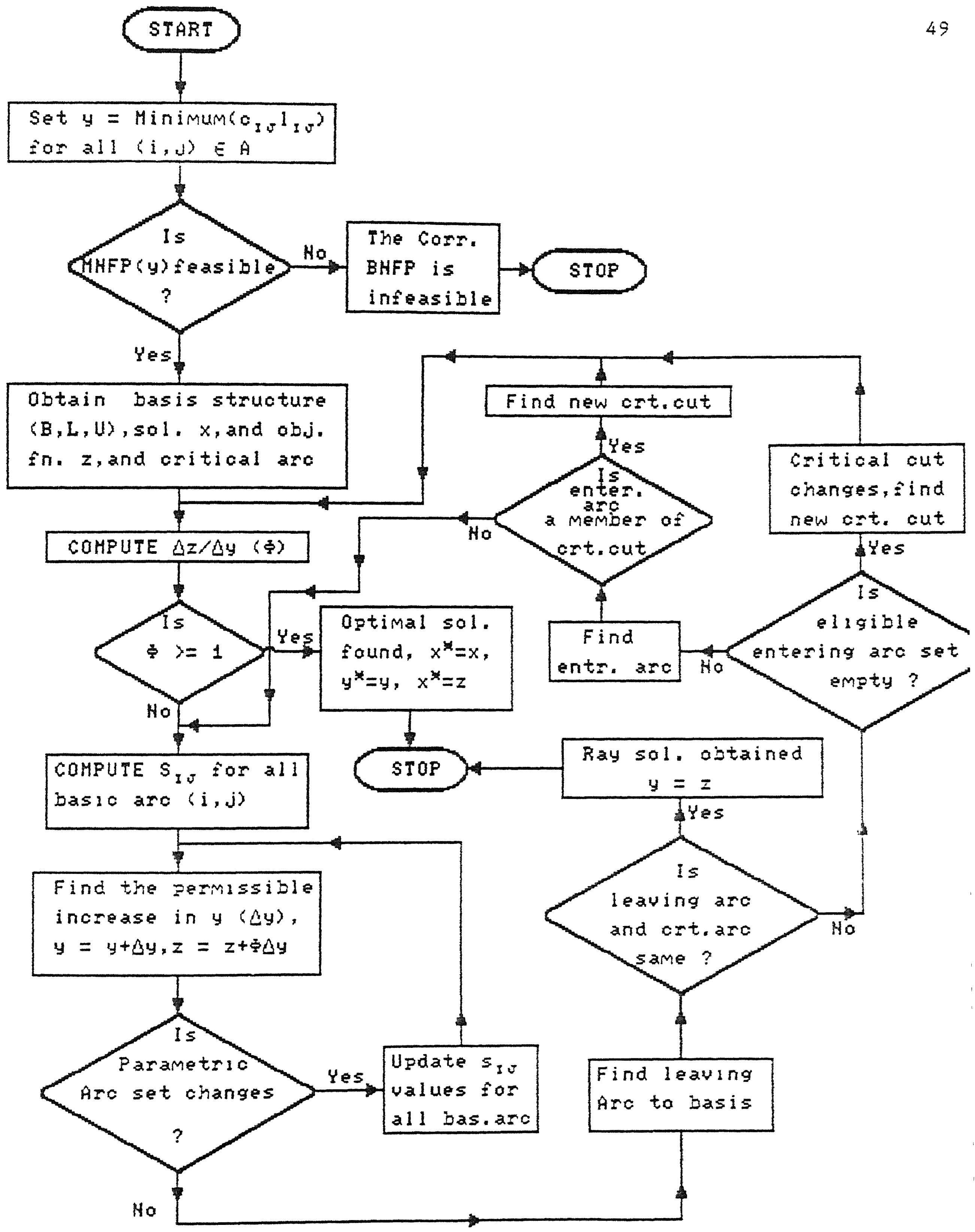

START
Set y = Minimum($c_{IJ}l_{IJ}$) for all (i,j) ∈ A
Is MNFP(y)feasible ?
No
The Corr. BNFP is infeasible
STOP
Yes
Obtain basis structure (B,L,U),sol. x,and obj. fn. z,and critical arc
COMPUTE Δz/Δy (Φ)
Is Φ >= 1
Yes
Optimal sol. found, x*=x, y*=y, x*=z
STOP
No
COMPUTE S_{IJ} for all basic arc (i,j)
Find the permissible increase in y (Δy), y = y+Δy,z = z+ΦΔy
Is Parametric Arc set changes ?
Yes
Update s_{IJ} values for all bas.arc
No
Find leaving Arc to basis
Is leaving arc and crt.arc same ?
Yes
Ray sol. obtained y = z
STOP
No
Is eligible entering arc set empty ?
Yes
Critical cut changes,find new crt. cut
No
Find entr. arc
Is enter. arc a member of crt.cut
Yes
Find new crt.cut
No

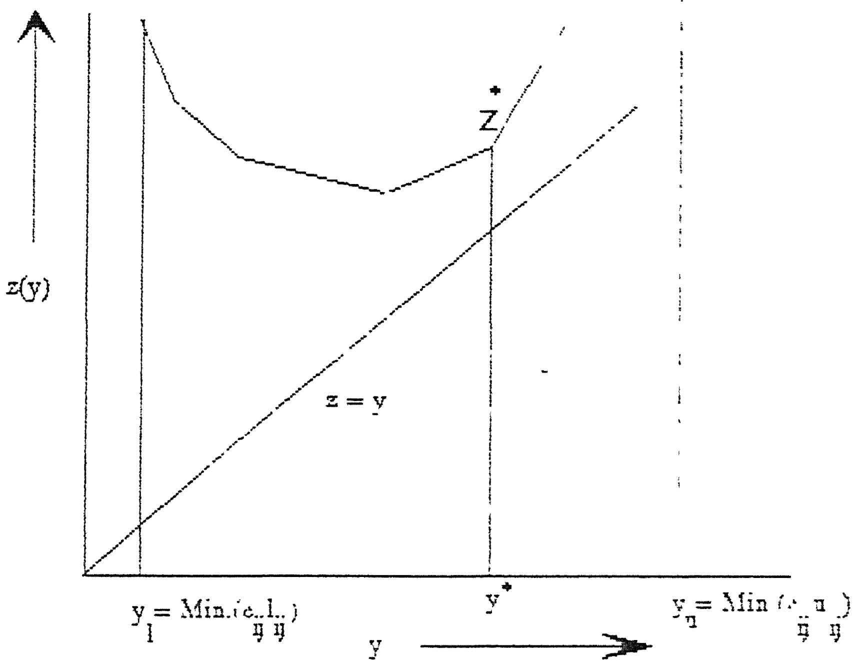
Figure 3.1 Flow Chart of Parametric Algorithm for BNFP

This may not be the case in many applications as one would like to balance weighted arc flows in a subset A' of arcs with $A' \leq A$. The following modification while finding the leaving arc will take care of such cases. Let us denote $\tilde{A} = A - A'$. Set $\Delta y^1_{ij} = \infty$ and $\Delta y^2_{ij} = \infty$ for $(i,j) \in \tilde{A}$. This modification will work, because the parametric upper bound and the parametric lower bound conditions will not be checked for arcs in $\tilde{A}$, whereas the real lower bound and upper bound conditions will restrict flows of arcs to remain within their respective bounds.

3.3 BINARY SEARCH ALGORITHM

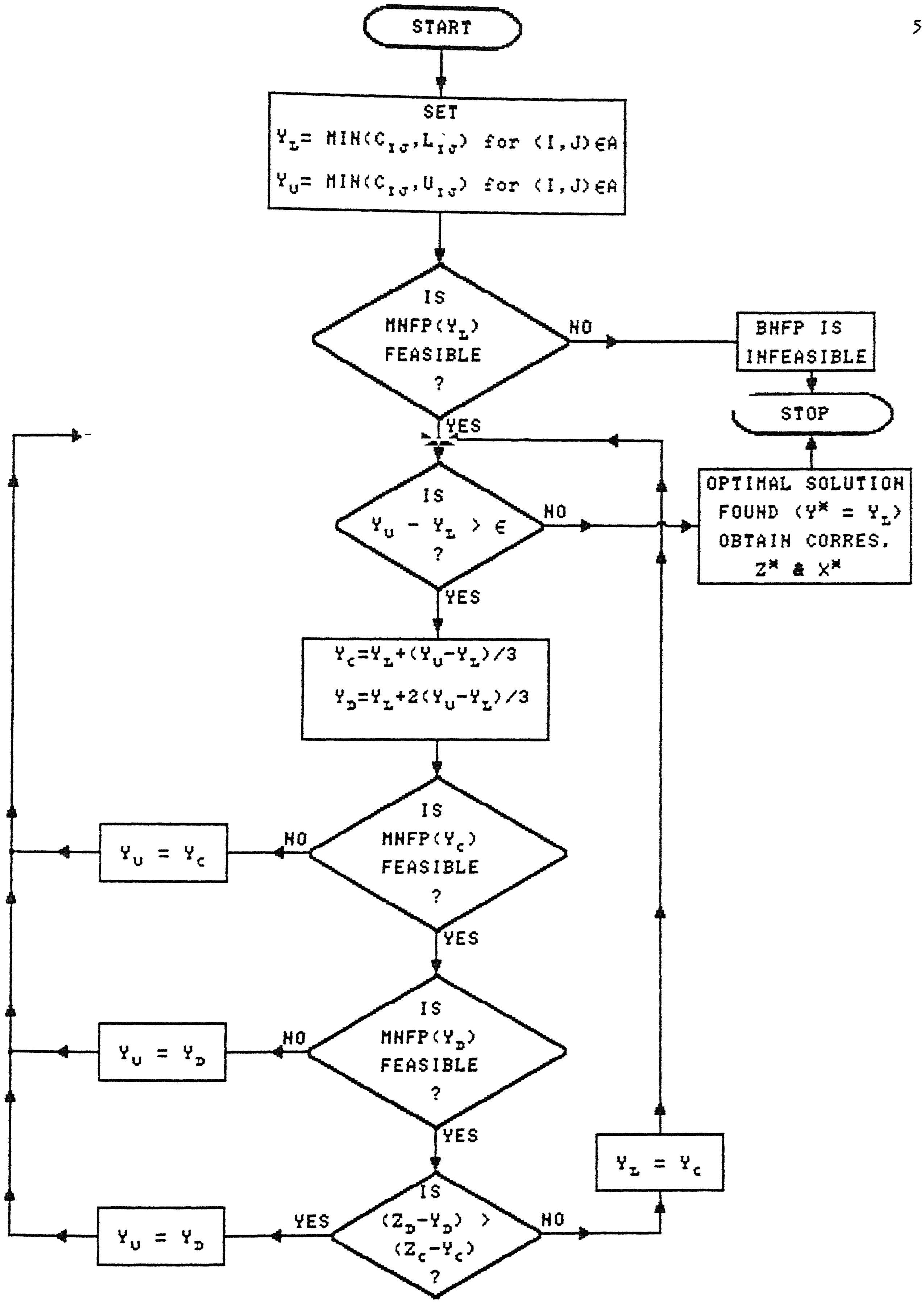
If we represent y = Minimum($c_{ij}x_{ij}$) $\forall$ $(i,j) \in A$ and z = Maximum($c_{ij}x_{ij}$) $\forall$ $(i,j) \in A$, then for a fixed value of y, say y_1, we can determine the corresponding minimum z value, say z_1, by solving a MNFP(y_1). Let us denote $z(y)$ as the objective function value of MNFP(y). As per Proposition 3.2, $z(y)$ is a piecewise linear convex function. The typical form of the $z(y)$ can be depicted by Figure 3.2. In the BNFP, we are interested in finding a solution x^*, with $y = y^*$ and corresponding $z(y) = z^*$ such that, the difference $(z^* - y^*)$ is minimized. The binary search algorithm finds y^* value by using the binary search technique in an interval of y. The interval is initialized as the lower and upper bound values of y, and is determined from the capacity constraints as below:

$$l_{ij} \leq x_{ij} \leq u_{ij} \ \forall \ (i,j) \in A$$



Showing Relationship Between Objective Function(z) of MNFP(y) and y

Figure 3.2



START
SET
Y_L= MIN(C_{IJ},L_{IJ}) for (I,J)∈A
Y_U= MIN(C_{IJ},U_{IJ}) for (I,J)∈A
IS MNFP(Y_L) FEASIBLE ?
NO
BNFP IS INFEASIBLE
STOP
YES
IS Y_U - Y_L > ∈ ?
NO
OPTIMAL SOLUTION FOUND (Y^* = Y_L) OBTAIN CORRES. Z^* & X^*
YES
Y_C=Y_L+(Y_U-Y_L)/3
Y_D=Y_L+2(Y_U-Y_L)/3
IS MNFP(Y_C) FEASIBLE ?
NO
Y_U = Y_C
YES
IS MNFP(Y_D) FEASIBLE ?
NO
Y_U = Y_D
YES
IS (Z_D-Y_D) > (Z_C-Y_C) ?
YES
Y_U = Y_D
NO
Y_L = Y_C

Figure 3.3 Flow Chart For Binary Search Algorithm

$\Leftrightarrow c_{ij}l_{ij} \le c_{ij}x_{ij} \le c_{ij}u_{ij} \;\forall\; (i,j) \in A$

$\Leftrightarrow \text{Minimum}(c_{ij}l_{ij}) \le \text{Minimum}(c_{ij}x_{ij}) \le \text{Minimum}(c_{ij}u_{ij})$

$\Leftrightarrow \text{Minimum}(c_{ij}l_{ij}) \le y \le \text{Minimum}(c_{ij}u_{ij})$

The binary search algorithm continuously improves the lower bound or the upper bound in each iteration, reducing the search interval. Theoretically, the algorithm will never terminate, since the interval will not merge to a point. In order to prevent this, a very small interval ε is to be set at the beginning, so that whenever the search interval becomes smaller than ε, the algorithm stops. Thus the binary search algorithm does not give an exact solution but gives a solution, which is within a small range of the optimal solution.

ALGORITHM STATEMENT

A stepwise description of the binary search algorithm is given below.

Step 1 Set $y_l = \text{Minimum}(c_{ij}l_{ij}) \;\forall\; (i,j) \in A$, $y_u = \text{Minimum}(c_{ij}u_{ij}) \;\forall\; (i,j) \in A$ and let ε be the desired interval for termination. Solve MNFP(y_l). If the problem is infeasible, then the corresponding BNFP is infeasible and *Stop*; else let x_l be the corresponding solution and z_l be the objective function value.

Step 2 If $(y_u - y_l) < \varepsilon$, then go to step 3; else compute the following

$$y_c = \frac{y_u - y_l}{3} + y_l$$

$$y_d = \frac{2(y_u - y_l)}{3} + y_l$$

Solve MNFP(y_c). If infeasible, then set $y_u = y_c$ and go to the beginning of step 2. Solve MNFP(y_d). If infeasible, then set $y_u = y_d$ and go to the beginning of step 2. Let z_c and z_d be the objective function values of MNFP(y_c) and MNFP(y_d) respectively. If $(z_d - y_d) > (z_c - y_c)$ then set $y_u = y_d$ else set $y_l = y_c$. Go to the beginning of step 2.

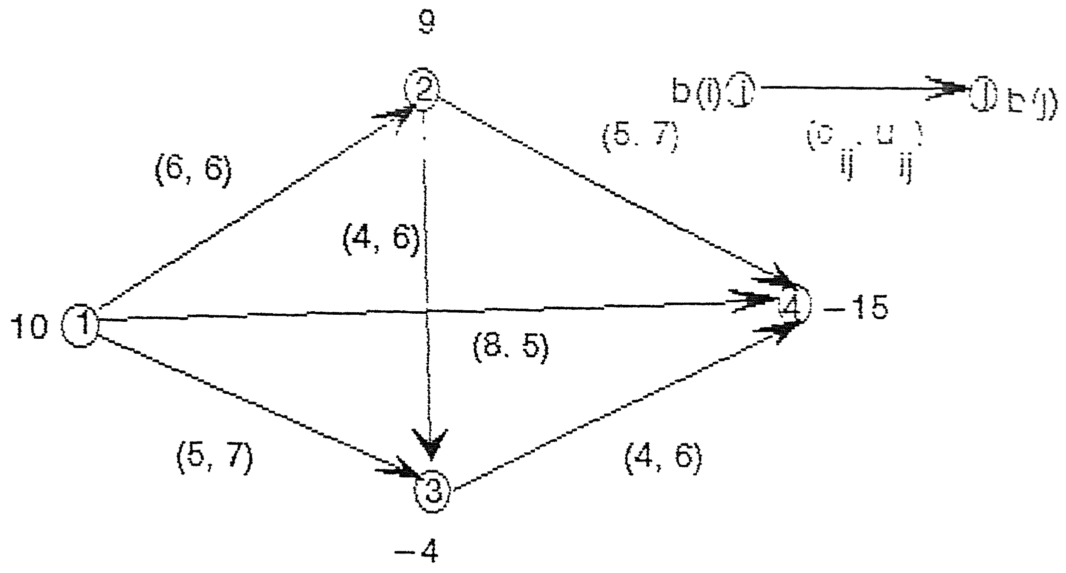
Step 3 $y^* = y_l$, $z^* = z_l$ and $x^* = x_l$. The objective function value of BNFP is $z^* - y^*$ and the solution is x^*. *Stop.*

The flow chart of the binary search algorithm is shown in Figure 3.3.

3.4 NUMERICAL EXAMPLE

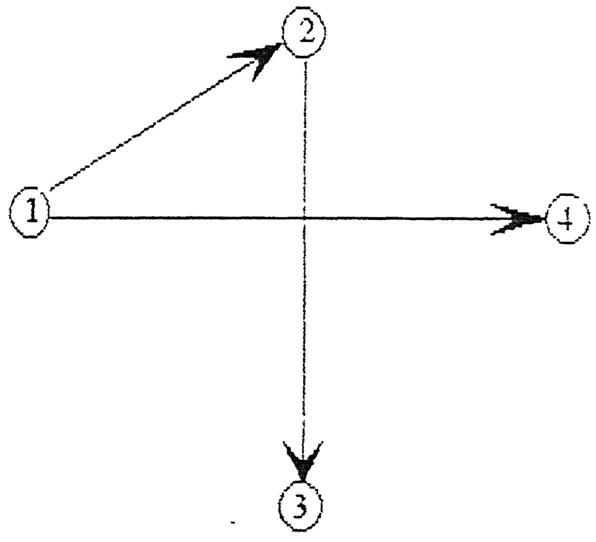
We solve a numerical example to illustrate various steps of the parametric algorithm. Consider the sample network shown in Figure 3.4. The first number besides each arc is its cost (weight) and the second number is its capacity. The label associated with each node is the supply/demand of that node. The lower bound on each arc is zero.

Since Minimum($c_{ij}l_{ij}$) = 0, we set y = 0. After solving



Numerical Example Problem

Figure 3.4



Initial Basis for the Example Problem

Figure 3.5

MNFP(y) and then transforming the solution to a spanning tree solution, we obtain a basis as shown in Figure 3.5. The basis structure and the objective function value 'z' of the solution is given below:

$B = \{(1,2),(1,4),(2,3)\}$, $L = \phi$, $U = \{(1,3),(2,4),(3,4)\}$,
$U_p = \{(1,3),(2,4)\}$

Since $l_{ij} = 0$, $L = L_p$ and $\Delta y^1_{ij} = \Delta y^3_{ij}$. The critical arc is (1,4), and it is at the parametric upper bound. The critical cut Q_{14} = $\{(1,4),(2,4),(3,4)\}$, and $z = 27.69$. The ratio $\theta = \Delta z/\Delta y = 0$. We tabulate the s_{ij} values of all basic arcs and the Δy_{ij} values in Table 3.1.

Table 3.1

Arc	s_{ij}	Δy^1_{ij}	Δy^2_{ij}	Δy^4_{ij}	Δy_{ij}	Δy^D
(1,2)	0.00	6.00	∞	∞	6.00	6.00
(2,3)	0.00	17.84	∞	∞	17.84	
(1,4)	0.00	27.69	∞	∞	27.69	

$\Delta y^u_{13} = \Delta y^u_{24} = \infty$, since $\theta = 0$. So $\Delta y = \text{Minimum}(\Delta y^D, \Delta y^u) = 6.00$, $\dot{y} = y + \Delta y = 6.00$, The basic arc (1,2) leaves the basis at its parametric lower bound. The eligible entering arcs set are $K = \{(1,3)\}$. Arc (1,3) enters to the basis. The new basis is as shown in Figure 3.6. The new basis structure is shown as below:

B = {(2,3),(1,4),(1,3)}, L = {(1,2)}, U = {(2,4),(3,4)}, U_p = {(2,4)}. The recomputed s_{ij} and Δy_{ij} are shown in Table 3.2.

Table 3.2

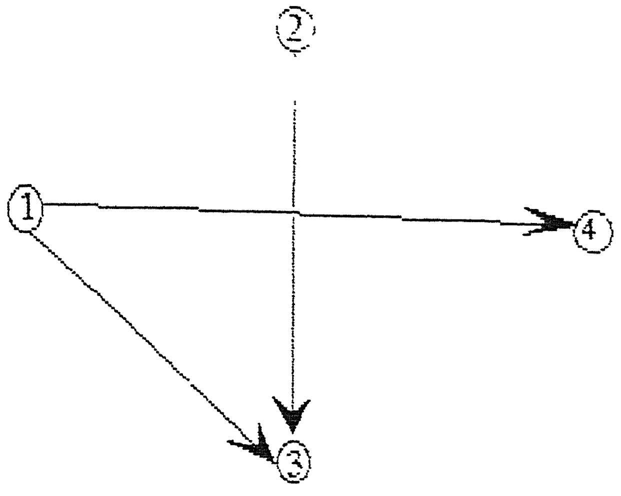
Arc	s_{ij}	Δy^1_{ij}	Δy^2_{ij}	Δy^4_{ij}	Δy_{ij}	Δy^b
(2,3)	0.00	35.53	14.76	9.23	9.23	9.23
(1,4)	0.00	21.69	∞	∞	21.69	
(1,3)	0.00	11.83	∞	∞	11.83	

$\Delta y = \Delta y^b = 9.23$, $y = y + \Delta y = 15.23$. The basic arc (2,3) leaves the basis at its upper bound. The eligible entering arc set K = ϕ. The critical cut changes now. The cut Q_{23} = {(1,2),(2,4),(3,4)}. Any parametric arc from Q_{23}, say (1,2), enters the basis as a critical arc. The updated basis is shown in Figure 3.7. The new basis structure is:

B = {(1,4),(1,3),(1,2)}, L = ϕ, U = {(2,4),(3,4),(2,3)}, U = {(2,4)}. New θ = 0.833. The recomputed s_{ij} and Δy_{ij} are shown in Table 3.3.

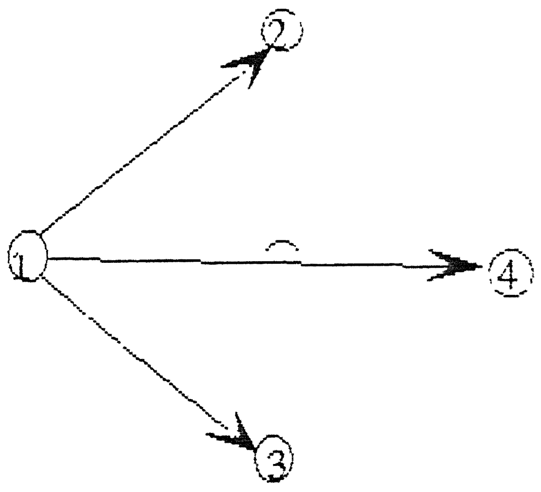
Table 3.3

Arc	s_{ij}	Δy^1_{ij}	Δy^2_{ij}	Δy^4_{ij}	Δy_{ij}	Δy^b
(1,4)	0.16	5.34	∞	∞	5.34	
(1,3)	0.00	4.77	∞	∞	4.77	4.77
(1,2)	-0.16	∞	74.77	20.77	20.77	



Basis After 1st Pivot

Figure 3.6



Basis After 2nd Pivot

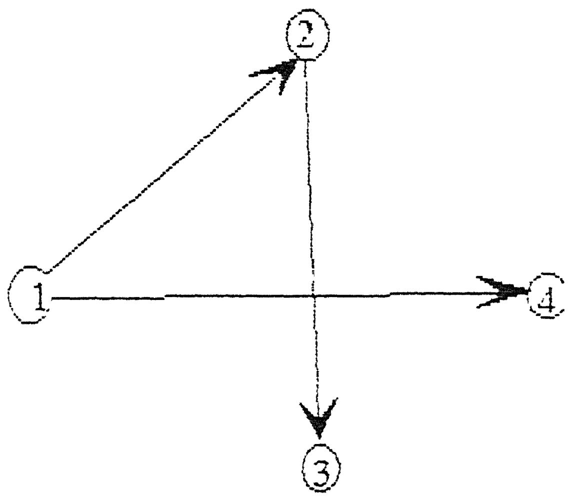
Figure 3.7

Δy^u_{24} =8.77. Δy = Minimum(Δy^b,Δy^u) = 4.77, y = y +Δy = 20.0, z = z + θ Δy = 31.66. The basic arc (1,3) leaves the basis at its parametric lower bound. The eligible entering arcs are K = {(2,3)}. Arc (2,3) enters to the basis, the new basis is as shown in Figure 3.8. The new basis structure is:
B = {(1,2),(1,4),(2,3)}, L = {(1,3)}, U = {(2,4),(3,4)}, U_p = {(2,4)}. The recomputed value of θ = 1.833, so the optimal solution is found. The solution is tabulated in Table 3.4.

Table 3.4

$z^* = 31.66,\ y^* = 20.00,\ z^* - y^* = 11.66$

Arc	Flow (x_{ij})	Cost(c_{ij})	u_{ij}	$c_{ij}x_{ij}$
(1,2)	3.33	6.00	6.00	20.00
(1,3)	4.00	5.00	7.00	20.00
(1,4)	2.67	8.00	5.00	21.33
(2,3)	6.00	4.00	6.00	24.00
(2,4)	6.33	5.00	7.00	31.67
(3,4)	6.00	4.00	6.00	24.00



Basis After 3rd Pivot

Figure 3.8

CHAPTER IV

IMPLEMENTATION DETAILS AND COMPUTATIONAL RESULTS

In this chapter, we describe various details required in the implementations of the parametric algorithm, the binary search algorithm and the linear programming approach for balanced network flow problem. We also present computational results of these algorithms.

4.1 SOLVING MINIMAX NETWORK FLOW PROBLEM (MNFP)

The parametric algorithm and the binary search algorithm as discussed in the previous chapter need a solution methodology to solve the minimax network flow problem, MNFP(y), where y is a parameter. Keeping the performance and simplicity of coding in mind, we extended the *Primal-Dual algorithm for minimax transportation problem* of Ahuja [23] for general networks.

PRIMAL-DUAL ALGORITHM FOR MNFP(Y)

The MNFP(y) defined over a network G = (N,A) is to solve the following optimization problem.

Minimize z

subject to

$$\sum_{j \in N} x_{ij} - \sum_{j \in N} x_{ji} = b(i), \ \forall \ i \in N,$$

$\text{Maximum}(l_{ij}, d_{ij}y) \le x_{ij} \le \text{Minimum}(u_{ij}, d_{ij}z)$, $\forall$ $(i,j) \in A$,

where $d_{ij} = 1/c_{ij}$ for all $(i,j) \in A$.

First of all, the above problem is transformed into a maximum flow problem with parametric upper bounds by adding a source node s and joining it with each supply node i by an additional arc (s,i) with capacity b(i). Similarly, a sink node t is added and each demand node j is connected with t by an additional arc (j,t) with capacity -b(j). We call these additional arcs as *artificial arcs* . Let $G' = (N', A')$ be the enlarged network. Now consider the following parametric maximum flow problem with z as a parameter.

Maximize v

subject to

$$\sum_{j \in N} x_{ij} - \sum_{j \in N} x_{ji} = \begin{cases} v & \text{if } i = s \\ 0 & \text{if } i \ne s,\ t \\ -v & \text{if } i = t \end{cases}$$

$\text{Maximum}(l_{ij}, d_{ij}y) \le x_{ij} \le \text{Minimum}(u_{ij}, d_{ij}z)$ $\forall$ $(i,j) \in A$,

$0 \le x_{si} \le b(i)$ $\forall$ $i \in N$ and $b(i) > 0$,

$0 \le x_{jt} \le -b(j)$ $\forall$ $j \in N$ and $b(j) < 0$,

The above problem is a maximum flow problem with parametric capacity. It is well known that the objective function value of a linear programming problem (maximization objective) with algorithm is given below.

Step 1 Compute $g_i = \dfrac{b(i)}{\sum_{j \in N} d_{ij}}$ for $i \in N$ such that $b(i) > 0$

$$h_j = \frac{-b(j)}{\sum_{j \in N} d_{ij}} \quad \text{for } j \in N \text{ such that } b(j) < 0$$

Set $k = 0$, $z^0 = \text{Maximum}\left\{ \underset{i\,:\,i \in N \text{ and } b(i)>0}{\text{Maximum}}(g_i),\ \underset{j\,:\,j \in N \text{ and } b(j)<0}{\text{Maximum}}(h_j) \right\}$

Step 2 Define capacity of each arc $(i,j) \in A$ as $d_{ij}z^k$ if $d_{ij}z^k <$ u_{ij} else u_{ij}. Similarly, define lower bound of each arc as the larger among $(d_{ij}y,\ l_{ij})$. Apply Maximum flow algorithm to determine the maximum flow in G' from source to sink. Let x^k_{ij} be the resulting flow and v^k be the maximum flow value. If $v^k = T$, go to step 4.

Step 3. Label those nodes in G' for which flow can be augmented from node s. Let $C^k = \{ (i,j) : (i,j) \in A', i$ is labeled and j is not labeled $\}$. Let $S = \{ (i,j) : (i,j) \in C^k \cap A$ and $d_{ij}z^k < u_{ij} \}$. Compute $w^k = \sum_{(i,j) \in S} d_{ij}$ and $z^{k+1} = z^k + (T - v^k)/w^k$, $k = k+1$ and go to step 1.

Step 4 (Optimal solution found) $z^* = z$, $x^* = x^k$ and *Stop.*

The above steps of the algorithm is represented by flow chart as shown in Figure 4.2. The primal-dual algorithm needs a

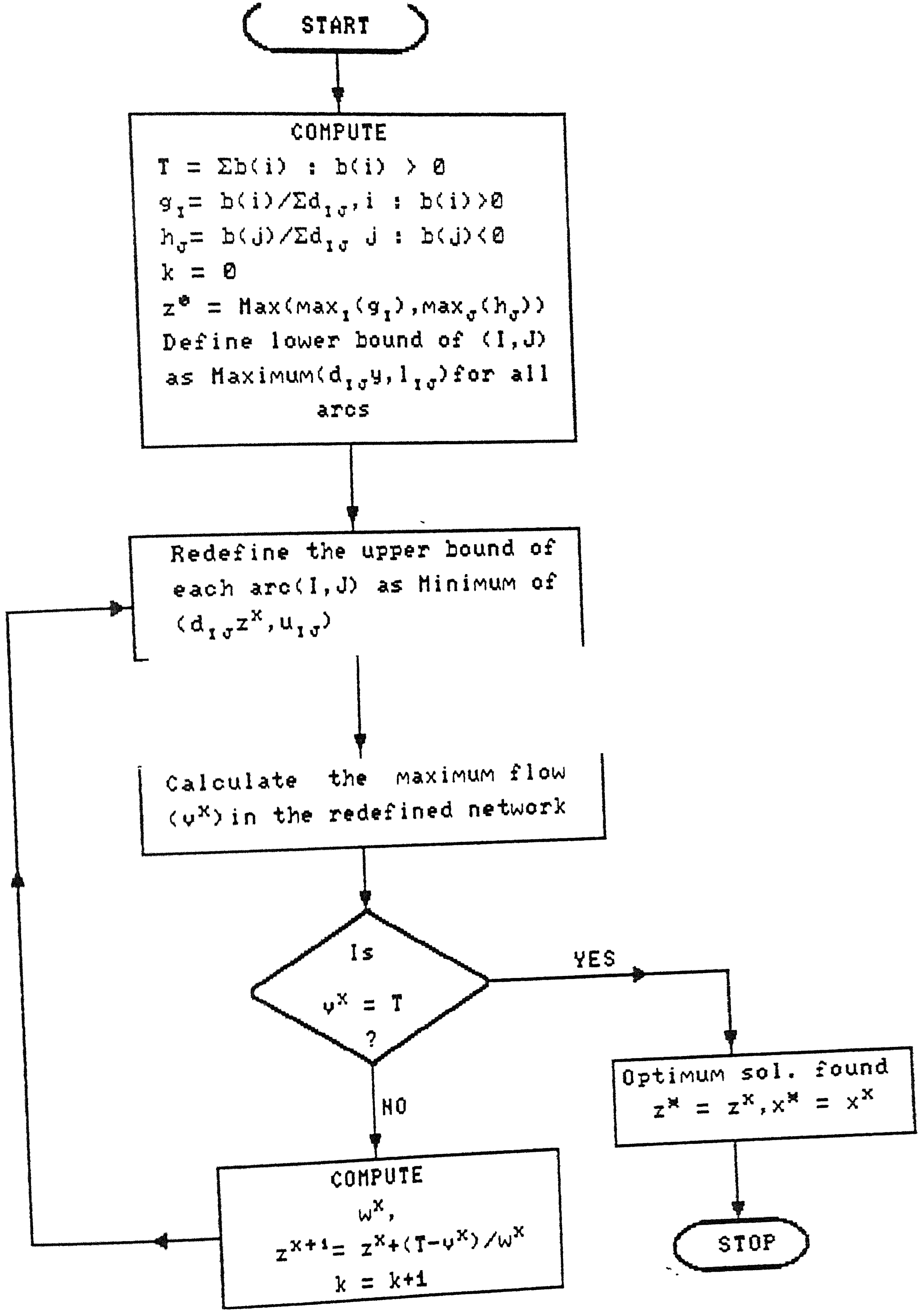

START
COMPUTE
T = Σb(i) : b(i) > 0
g_I = b(i)/Σd_IJ, i : b(i)>0
h_J = b(j)/Σd_IJ j : b(j)<0
k = 0
z° = Max(max_I(g_I), max_J(h_J))
Define lower bound of (I,J) as Maximum(d_IJ y, l_IJ) for all arcs
Redefine the upper bound of each arc(I,J) as Minimum of (d_IJ z^x, u_IJ)
Calculate the maximum flow (v^x) in the redefined network
Is v^x = T ?
YES
NO
Optimum sol. found z* = z^x, x* = x^x
STOP
COMPUTE
w^x,
z^(x+1) = z^x + (T − v^x)/w^x
k = k+1

Figure 4.2 Flow Chart of Primal-Dual Algorithm for MNFP(y)

maximum flow subroutine to solve the MNFP(y), and its computational performance is determined by the efficiency of the maximum flow code. We use a version of maximum flow subroutine which is based on *shortest augmenting path algorithm.*

4.2 GETTING THE INITIAL BASIS AND CRITICAL CUT

The primal-dual algorithm, when terminates, does not yield a basic solution (extreme point solution). This is in fact true for most network optimization algorithms because most network flow problems do not possess a unique optimum solution. Rather, they have alternate optimum solutions. We need to convert the optimum solution obtained from primal-dual algorithm to an extreme point solution, i.e., a cycle free solution in the network optimization language. We use the following steps to obtain a cycle free solution.

Step 1 After solving the MNFP(y), add all arcs at their upper bounds (either parametric or real) to set U. Add all arcs at their lower bounds (parametric or real) to set L, and the remaining arcs to set B.

Step 2 If B does not contain a cycle, go to step 3; else identify a cycle in set B and then remove the cycle from B by augmenting flow in clockwise (or anti-clockwise) direction until one arc in the cycle reaches its upper bound (parametric or real) or its lower bound (parametric or real). Delete the arc from B and add it to U or L depending on its bound. Repeat this step.

Step 3 A cycle free solution has been found. *Stop.*

At the end of the above steps we get a cycle free solution, i.e., the set B consists of arcs which form a forest (union of trees). The trees in B are disconnected by cuts. We now require to identify a critical cut among these cuts and add some restricted arcs between these cuts so as to get a basis which is a spanning tree. We use the following steps to get a spanning tree basis and identify a critical cut. The following procedure partitions the set N of node into two parts: marked nodes and unmarked nodes.

Step 1. Make all nodes in N unmarked. Select any node from N, say i, and mark it.

Step 2 Use *search algorithm* to identify all nodes which are reachable from i. Make all these nodes marked.

Step 3 If there is no unmarked node, then go to step 5. If critical cut is yet to be found, then check the optimality condition of Theorem 3.2 for the cut C(marked, unmarked). If it satisfies, then it is the critical cut.

Step 4 If C(marked, unmarked) is the critical cut, then add a parametric arc ; else add any arc from this cut to the basis. In the former case the arc is called critical arc. Go to step 2.

Step 5 A critical cut and a spanning tree basis has been obtained. *Stop.*

We use *breadth first search* version of the search algorithm to perform various operations like identifying a cycle, finding a cut. The basic iterative steps of the breadth first search procedure is described as follows.

```
Algorithm breadth first search;
begin
     unmark all nodes in N;
     select any node i;
     mark node i; pred(i) = 0;
     Queue = {i};
     While Queue ≠ φ do
     begin
          select first element of the queue;
          if node i is incident to arc (i,j) and j ≠ pred(i) then
          begin
               mark node j;
               pred(j) = i;
               add node j to Queue;
          end;
```

```
        else delete node i from Queue;
    end;
end;
```

4.3 NETWORK REPRESENTATION

Network representation is one important factor that significantly affects the running time of an algorithm. A clever network representation not only reduces the storage requirement, but also facilitates easy updating, and easy maintenance of the intermediate results. We use the forward star representation to store networks as it suits the requirements of our algorithm. We describe below this representation.

In the forward star representation, we arrange the arcs in A in certain order: we first arrange arcs emanating from node 1, then from node 2, and so on. Arcs emanating from the same node can be numbered arbitrarily. We then sequentially store their tail, head, cost and capacity in that order. We maintain a pointer with each node i, denoted by point(i), that indicates the smallest number of the arc emanating from node i. Hence the outgoing arcs from node i are stored at position point(i) to point(i+1)-1 in the arc list. If point(i) > point(i+1) - 1, then node i has no outgoing arc. For consistency, we set point(1) = 1 and point(n+1) = m+1. We illustrate in the Table 4.1 forward star representation of network already shown in Figure 3.4.

Table 4.1

Arc no.	Point	Arc	Cost	Capacity
1	1	(1,2)	6.0	6.0
2	4	(1,3)	5.0	7.0
3	6	(1,4)	4.0	6.0
4	7	(2,3)	4.0	6.0
5	7	(2,4)	5.0	7.0
6		(3,4)	4.0	6.0

4.4 DATA STRUCTURE USED

The parametric algorithm maintains a basis at every iteration. While performing various operations on the basis, like identifying a cycle, finding a cut, we require access to all arcs incident to a node. We use the following data structure to maintain the basis and perform above operations efficiently. We use an array, called bstar, which arranges the arc numbers of arcs in the basis in certain order. It first arranges all arcs incident to node 1, then node 2, and so on. It stores same arcs in two places because it is incident to two nodes. Similar to the forward star representation, we keep a pointer to each node in a array, called bpoint, which gives starting position of arcs emanating from a node. For easy updating of the basis, we keep all basic arcs in another array, called bas, and keep track of the leaving arc from a basis by maintaining a logical array, called bdelet, whose i-th element is made true if arc i leaves the basis. We modify the bstar and bpoint array whenever the basis changes.

4.5 IMPLEMENTATION OF LINEAR PROGRAMMING CODE

We compared the execution times of parametric and binary search algorithms with the simplex algorithm applied to the enlarged linear programming problem. We used the XMP library developed by Roy E. Marsten [32] for solving the enlarged linear program. The XMP library contains subroutines written in VANILLA-FORTRAN to solve different sizes of linear programs. We used those subroutines which are based on the *primal simplex algorithm* and exploit the sparseness of the constraint matrix. The number of eligible nonbasic variables which are put in the candidate list and the refactorization frequency of the basis which have an important bearing on the performance of the code were kept 6 and 50 respectively. The above values were set so that the LP code gave a better performance.

4.6 CHOICE OF NETWORK GENERATOR

For large scale computational testing, we need a network generator to generate random networks which are possibly good representation of reality. We used *Netgen*, the network generator program developed by Klingman, Napier and Stutz in 1974 [33]. Netgen can generate different sizes of minimum cost flow, transportation and assignment networks.

We generated the arc capacities, costs, total supply in the following manner. The capacity of an arc was generated as a random number in the interval [0.5n, 2.0n], cost in the interval [1.0, m], and the total supply is set equal to 50n, where n and m are number of nodes and number of arcs in the network respectively. We set the total number of source node equal to 0.2n, the total number of sink node equal to 0.25n. The total number of transshipment source nodes and transshipment sink nodes were set half of the total source and total sink nodes respectively. The percentage of skeleton arcs which were given maximum cost was 40% and the percentage of capacitated arcs was 40%.

4.7 COMPUTATIONAL RESULTS

The parametric algorithm, and the binary search algorithm were coded in FORTRAN-77. We also used the FORTRAN codes for the primal simplex algorithm of XMP library. The three codes were extensively tested on HP 9000/850s computer systems running under HP-UX operating system. We tested the codes with different sizes of problems generated by netgen. We considered seven sizes of problem and for each problem size we solved five problems. We noted down the CPU time and the following counts for each code.

For parametric algorithm :

1) Time to get initial basis,

2) Time for parameterization,

3) Number of pivot operations.

For binary search algorithm :

1) Number of search points

2) Number of calls to the maximum flow routine

3) Total number of flow augmentations

4) Total time to obtain the solution

For primal simplex algorithm :

1) Total number of pivot operations

2) Total time to obtain the solution

We have tabulated these values, averaged over five problems and presented in Tables 4.2 to 4.4. All time information are given in seconds. For comparison purposes we have given the CPU times of all the three codes in Table 4.5. For better insight of the data, the CPU time verses problem size plot for all the three codes have been shown in Figure 4.3 to 4.5. The average number of iterations verses problem size plot for parametric algorithm is shown in Figure 4.6. Using the regression analysis, we have estimated the time and number of iterations the parametric code will take for different problem sizes. The ratio of estimated time and actual time, and estimated iterations and actual number of iterations for different problem sizes have been shown in Figures 4.7 to 4.12.

Table 4.2

Computational Results for Parametric Algorithm

No. of Nodes	No. of Arcs	No. of Iterations	Minimum Time	Maximum Time	Time to Get basis	Time for Parameterization	Total Time
10	20	1.20	0.01	0.03	0.016	0.040	0.02
10	50	3.20	0.04	0.06	0.028	0.022	0.05
10	72	3.00	0.03	0.12	0.032	0.028	0.06
25	50	1.4	0.05	0.07	0.048	0.012	0.06
25	125	3.4	0.10	0.27	0.072	0.078	0.15
25	250	4.6	0.13	0.51	0.166	0.134	0.30
50	100	3.0	0.13	0.31	0.132	0.058	0.19
50	250	12.8	0.47	1.03	0.280	0.460	0.74
50	500	12.2	0.93	2.04	0.538	0.792	1.33
100	200	3.0	0.44	0.72	0.408	0.102	0.51
100	500	6.2	0.69	1.71	0.710	0.430	1.14
100	1000	22.8	3.86	5.56	1.664	2.846	4.51
200	400	3.6	1.38	2.17	1.418	0.262	1.68
200	1000	31.4	3.03	12.19	3.408	4.352	7.76
200	2000	22.8	6.25	23.64	5.998	5.962	11.96
500	1000	6.4	7.56	10.26	7.308	1.182	8.49
500	2000	72.0	20.58	44.49	15.506	22.314	37.82
500	5000	110.8	40.36	140.07	42.182	71.658	113.84
1000	2000	79.6	26.05	78.39	25.476	30.924	56.40
1000	4000	146.6	67.61	189.45	63.236	92.674	155.91
1000	10000	248.6	467.34	529.16	164.482	330.618	495.01

Table 4.3

Computational Results for Binary Search Algorithm

No.of Nodes	No.of Arcs	Minimum Time	Maximum Time	Average Time	No.of search points	No.of call to max.flow	No.of flow Augmentations
10	20	0.06	0.65	0.26	25.4	95.4	735.4
10	50	0.58	0.87	0.71	42.4	150.8	1914.8
10	72	1.01	1.41	1.12	166.6	3465.0	3465.0
25	50	0.63	1.85	1.22	32.0	172.4	3891.2
25	125	2.04	2.91	2.50	46.2	212.6	7646.4
25	250	3.10	5.02	3.83	45.6	210.0	0172.6
50	100	1.65	5.07	2.99	41.8	225.8	9871.8
50	250	6.08	7.68	6.72	49.8	297.2	8773.2
50	500	8.52	12.89	10.49	46.6	261.4	8256.4
100	200	4.40	13.19	6.73	29.8	189.4	18124.2
100	500	12.86	20.97	16.37	52.2	337.6	6139.8
100	1000	21.33	34.82	27.60	52.8	362.0	4046.8
200	400	9.87	16.42	12.99	31.2	219.2	37216.4
200	1000	23.77	42.43	36.18	52.6	393.0	100610.4
200	2000	52.82	78.35	60.10	51.2	399.8	148511.0
500	1000	32.43	58.04	44.27	35.8	283.4	114264.0
500	2000	74.18	119.17	99.34	48.60	423.8	58173.6
500	5000	132.61	218.12	171.24	47.4	426.4	377655.8
1000	2000	130.88	164.42	146.00	47.2	421.8	376816.0
1000	4000	188.58	220.37	204.49	51.0	376.4	511092.4
1000	10000	355.30	466.11	401.70	51.6	504.2	843066.2

Table 4.4

Computational Results of Primal Simplex Algorithm

No.of Nodes	No.of Arcs	Minimum Time	Maximum Time	Average Time	No.of Pivots
10	20	0.27	0.40	0.34	31.80
10	50	1.28	1.80	1.49	81.20
10	72	2.57	3.17	2.89	116.90
25	50	1.83	2.49	2.04	107.60
25	125	6.37	10.03	7.83	221.00
25	250	12.21	27.39	20.61	321.40
50	100	6.97	7.88	7.33	235.80
50	250	20.94	30.80	26.90	413.75
50	500	88.53	101.37	94.95	766.00
100	200	10.90	15.14	12.91	252.50
100	500	30.33	31.68	31.25	250.00*
100	1000	117.78	209.44	153.11	601.00
200	400	25.98	28.32	27.28	250.00*
200	1000	74.03	78.58	76.22	250.00*
200	2000	442.73	606.93	524.46	1015.00

Note * Optimal solution could not found due to large number of degenerate pivots.

Table 4.5

Comparison of Execution Times

No.of Nodes	No.of Arcs	Average Time		
		Parametric	B.search	Simplex
10	20	0.02	0.26	0.34
10	50	0.05	0.71	1.49
10	72	0.06	1.12	2.89
25	50	0.06	1.22	2.04
25	125	0.15	2.50	7.83
25	250	0.30	3.83	20.61
50	100	0.19	2.99	7.33
50	250	0.74	6.72	26.90
50	500	1.33	10.49	94.95
100	200	0.51	6.73	12.91
100	500	1.14	16.37	31.25*
100	1000	4.51	27.60	153.11
200	400	1.68	12.99	27.28*
200	1000	7.76	36.18	76.22*
200	2000	11.96	60.10	524.46
500	1000	8.49	44.27	
500	2000	37.82	99.34	
500	5000	113.84	171.24	
1000	2000	56.40	146.00	
1000	4000	155.91	204.49	
1000	10000	495.01	401.70	

Note * Optimal solution could not found due to large number of degenerate pivots.

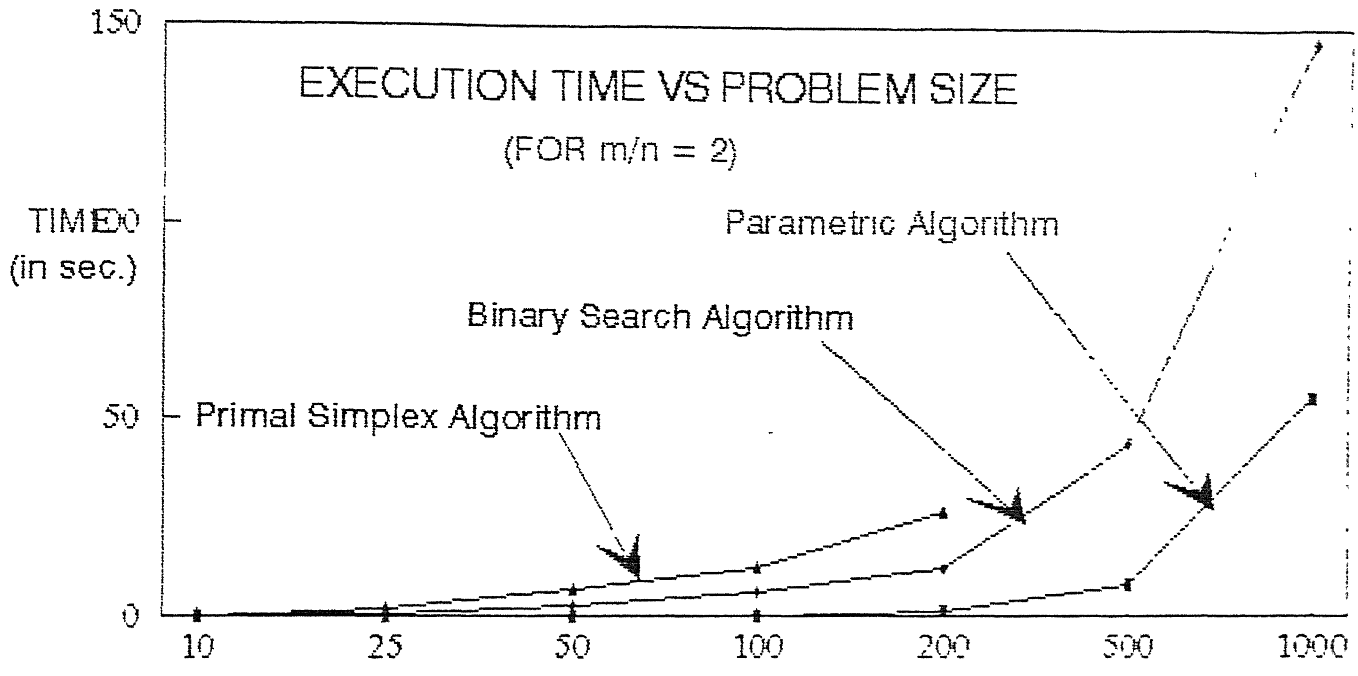


Figure 4.3

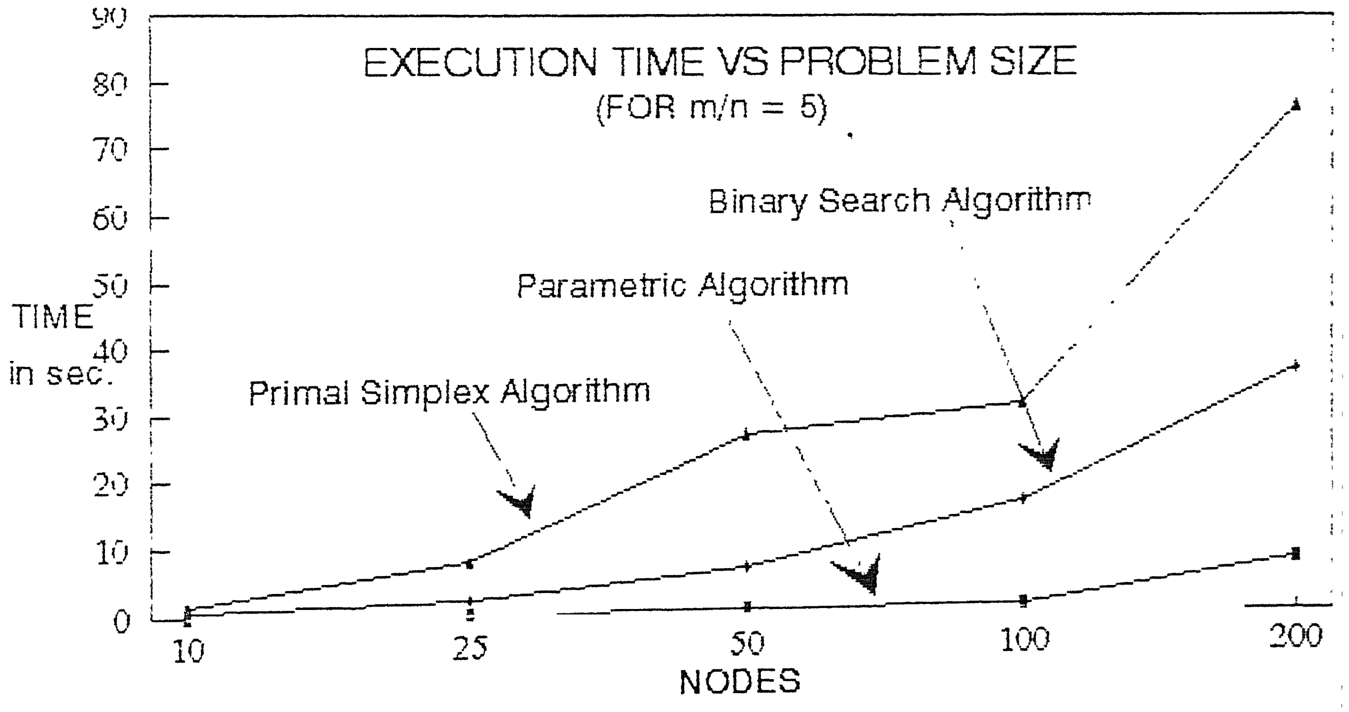
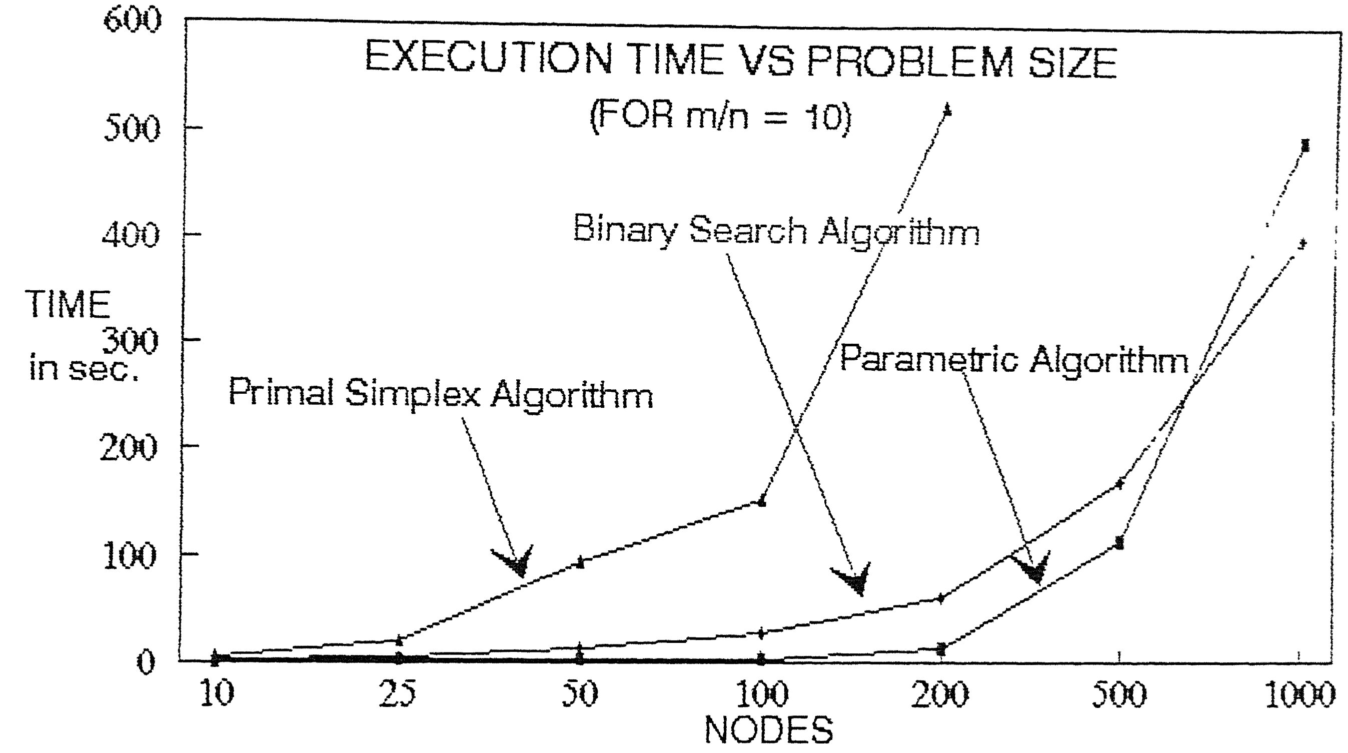
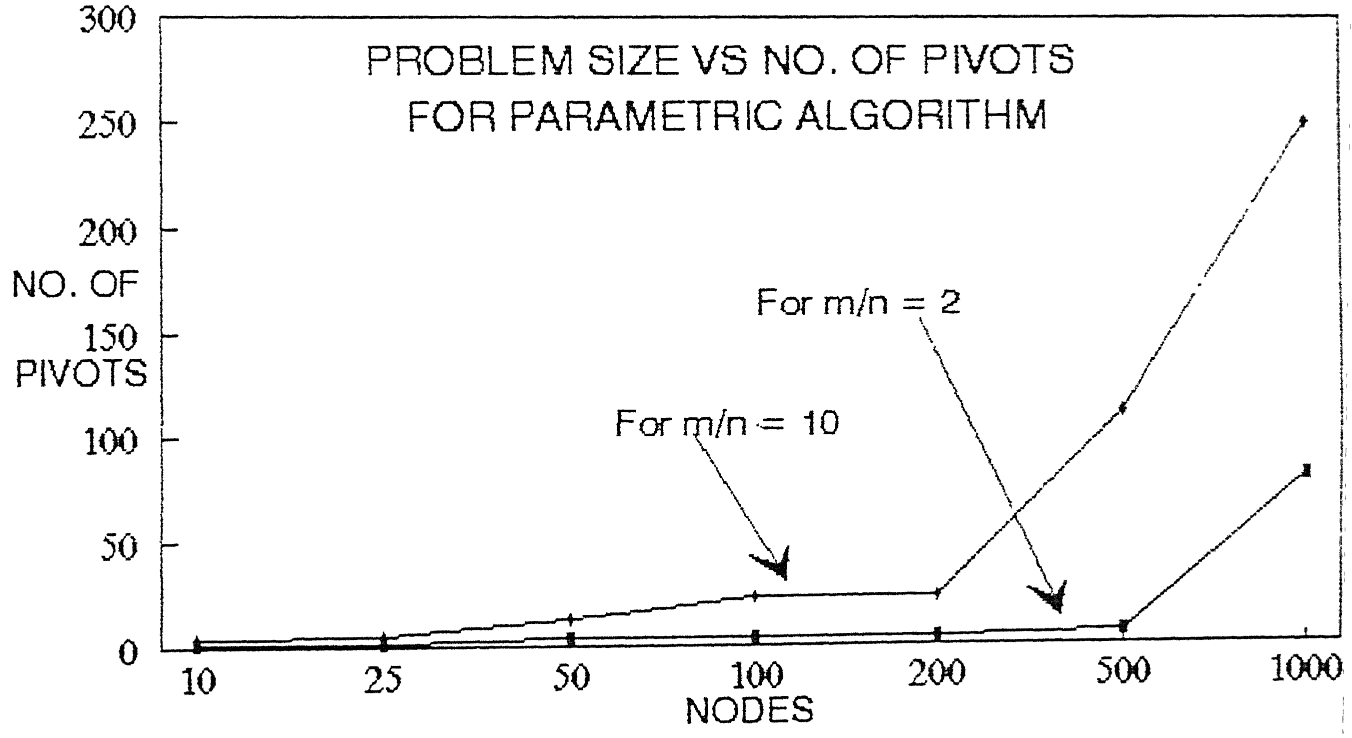


Figure 4.4



Fiqure 4.5



Fiqure 4.6

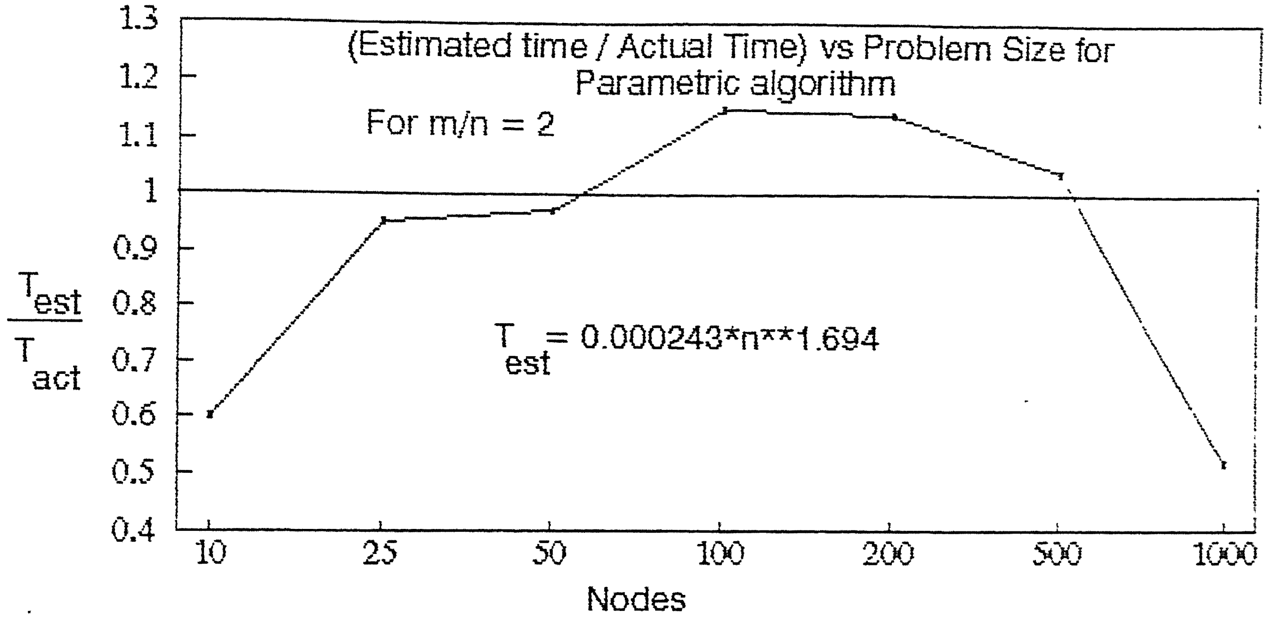


Figure 4.7

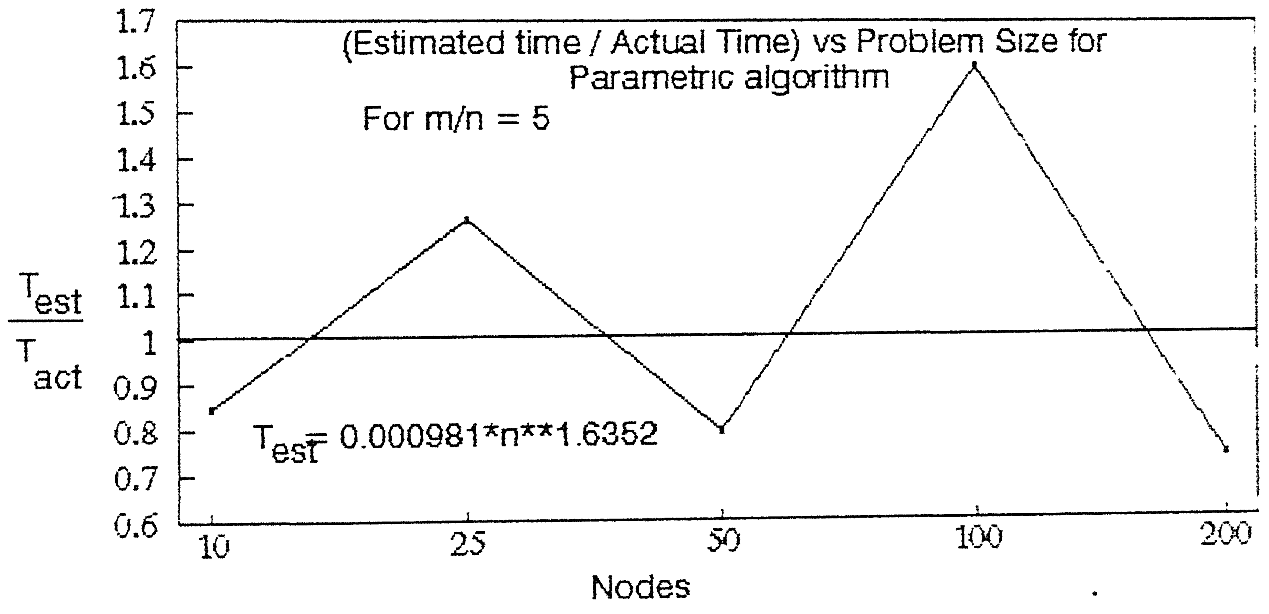


Figure 4.8

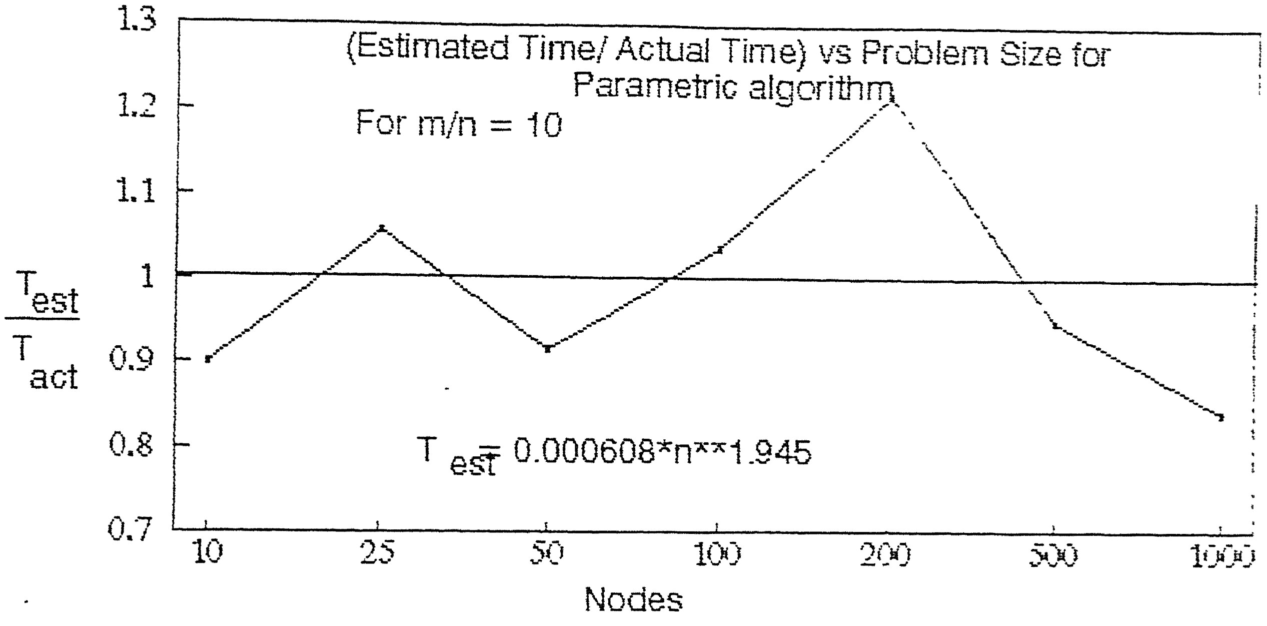


Figure 4.9

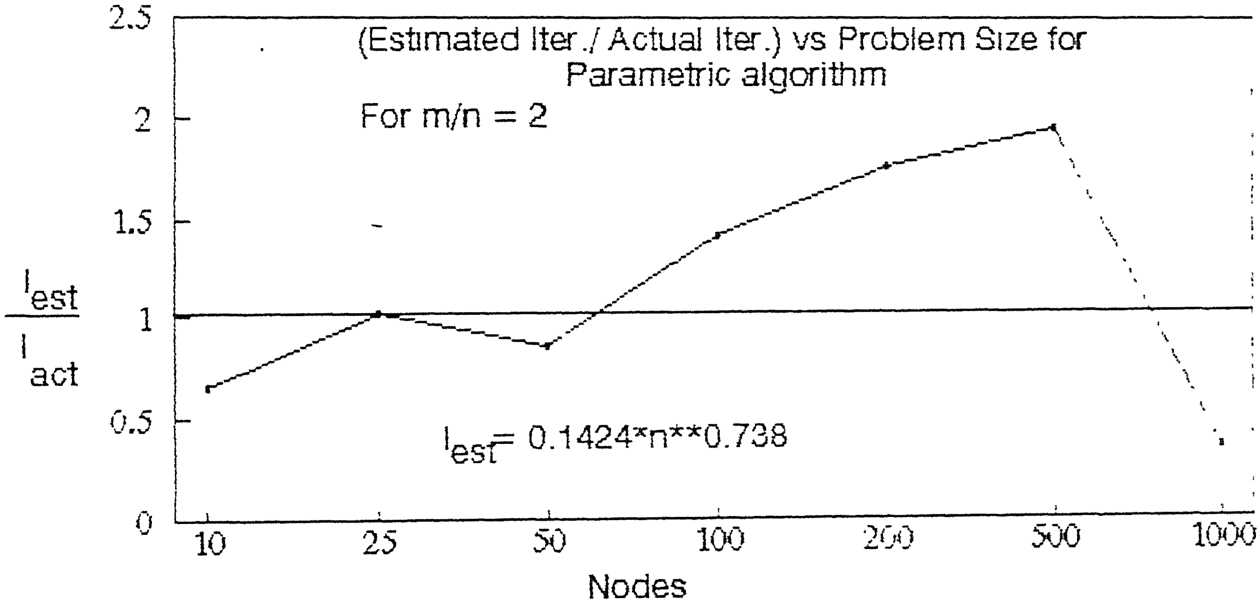


Figure 4.10

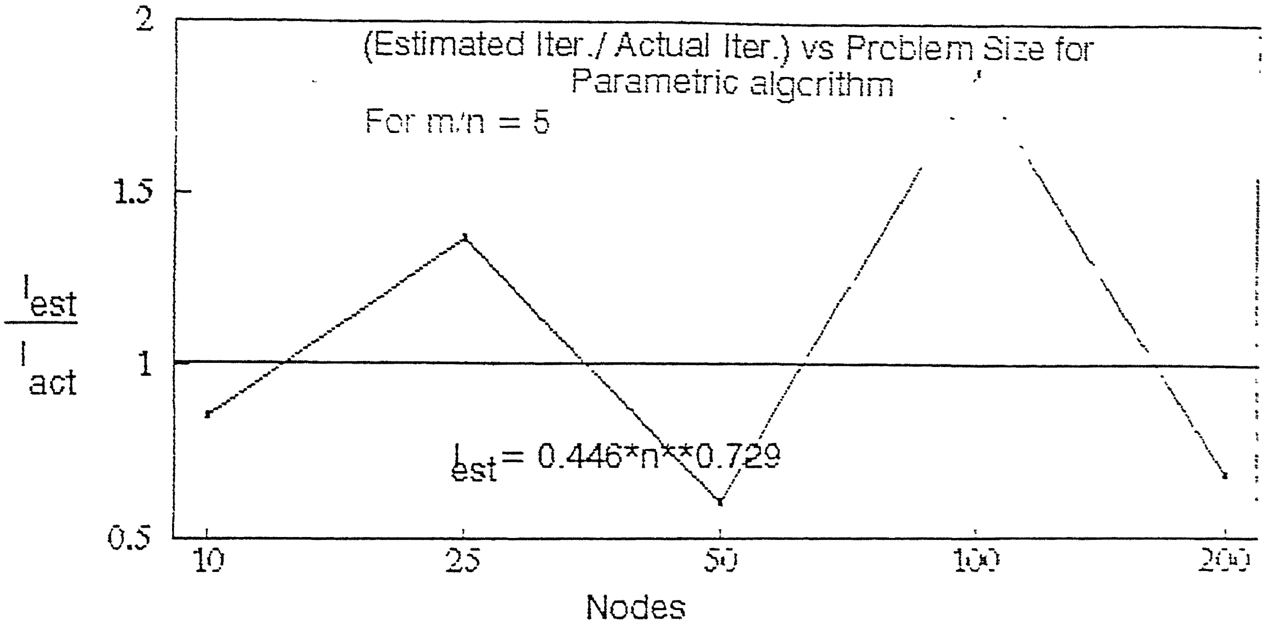


Figure 4.11

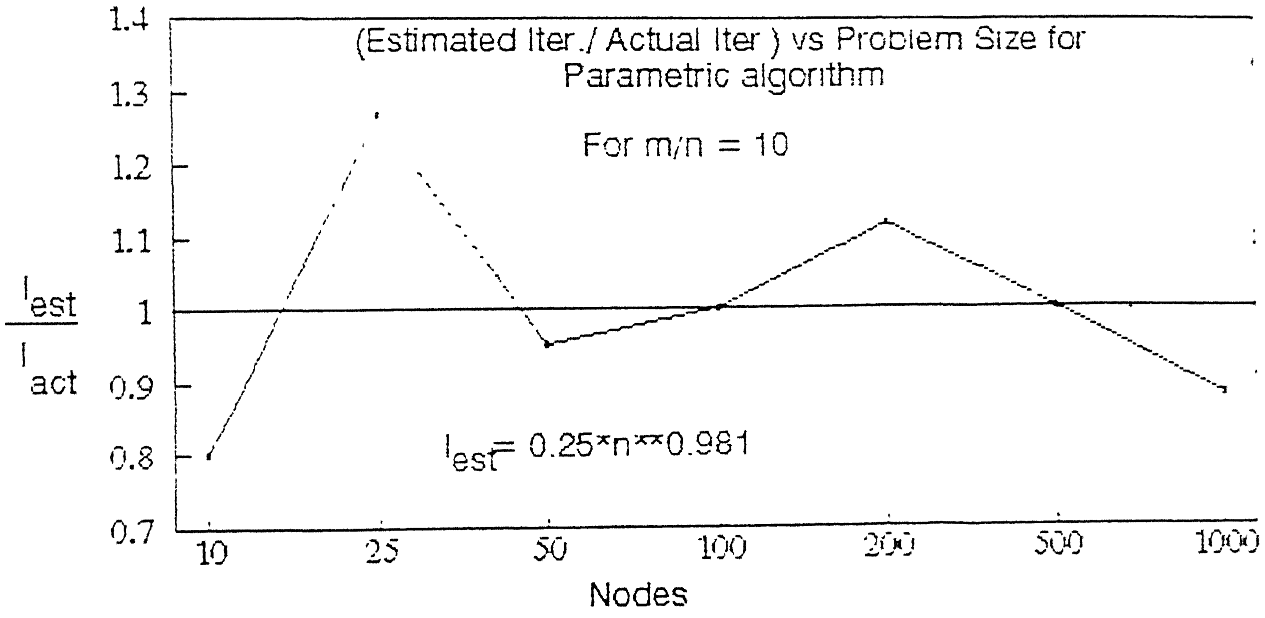


Figure 4.12

The data clearly shows the superiority of the parametric algorithm over binary search and primal simplex algorithms. The average number of iterations for the parametric algorithm is considerably smaller than that of the primal simplex algorithm. The parametric algorithm also requires less storage requirement than the primal simplex algorithm. While parametric algorithm requires $k_1m + k_2n$ storage, the simplex algorithm requires $k_3m^2 + k_4mn + k_5n^2$ storage, where $k_1, \ldots, k_5$ are some constants. We have also noted during testing that the primal simplex code could not get optimal solution for many problems due to the large number of degenerate steps. In fact, when the problem size became larger than 200 nodes and 2000 arcs, the simplex algorithm could not find the optimal solutions for most problems. For smaller size problem-s, the binary search and simplex code took large time compared to the parametric code. For large size problems the binary search code became competitive with the parametric code for higher m/n ratios. This is because of the following reason: as the problem size increases, the number of pivot operations also increases for parametric algorithm whereas the number of search points for the binary search algorithm does not change so much since it depends upon the maximum cost and maximum capacity of the arc set. we note that the binary search algorithm does not give an exact solution. Further, the parametric code can be made faster than our implement-ation by choosing better data structure, whereas the binary search code is more or less optimized.

4.8 CONCLUSIONS

In this thesis we have described two algorithms for the balanced network flow problem. We have specialized the parametric algorithm of balanced linear programming problem of Ahuja[30] for network problems. The parametric algorithm exploits the special structure of the constraint matrix to simplify the computations, easy updating and maintenance of the basis resulting in lesser CPU time. The binary search algorithm uses binary search technique to get near to the optimal solution. It does not give an exact solution but gives a solution which is within a specified interval around the optimal solution. The primal simplex algorithm to solve the balanced network flow problem is found to be uneconomical from execution time as well as storage point of view. Our computational testing shows the superiority of parametric algorithm over the other two methods.

REFERENCES

[1] J.R.Brown, "The Sharing Problem", **Operations Research, 27 pp 324-340, (1979).**

[2] S.Kaplan, "Applications of Programs with Maximin Objective Functions to Problem of Optimal Resource Allocation", **Operations Research 22, pp 802-907, (1974).**

[3] J.R.Brown, "The Linear Sharing Problem", **Operations Research 32, pp 1087-1105, (1984).**

[4] R.Gupta and S.R.Arora, "Programming Problem with Maximin Objective Function", **Opsearch 14, pp 125-130, (1977).**

[5] M.E.Posner and C.T.Wu, "A Note on Programming Problem with Maximin Objective Function", **Opsearch 15, pp 117-119, (1978).**

[6] S.Jacobsen, "On Marginal Allocation in Single Constrained Min-max Problems", **Management Science 17, pp 780-783, (1971).**

[7] E.L Porteous and J.S.Yonmark, "More on Min-max Allocation", **Management science 8, pp 502-507, (1972).**

[8] J.R.Brown "The Knapsack Sharing Problem", **Operations**

Research 27, pp 341-355, (1979).

[9] M.S.Bazaraa and J.J.Goode, "An Algorithm for Solving Linearly Constrained Minimax Problems", **European Journal of Operations Research 11, pp 158-166, (1982).**

[10] S.R.K.Datta and M.Vidyasagar, "New Algorithm for Constrained Minimax Optimization", **Mathematical Programming 13, pp 140-155, (1977).**

[11] R.K.Ahuja, "Minimax Linear Programming Problem", **Operatins Research Letters 4, pp 131-134, (1985).**

[12] J.R.Brown, "The Flow Circulation Sharing Problem", **Mathematical programming 25, pp 199-227, (1983).**

[13] D.F.Sanat and G.A.Mago, "Minimizing Maximum Flows in Linear Graphs", **Network 9, pp 333-361, (1979).**

[14] T.Ichimori, H.Ishii and T.Nishida, "Finding Weighted Minimax Flow in a polynomial Time", **Journal of Operations Research Society of Japan 23, pp 268-272, (1980).**

[15] T.Ichimori, H.Ishii and T.Nishida, "Weighted Minimax Real-Valued Flows", **Journal of Operations Research Society of Japan", pp 52-59, (1981).**

[16] N.Megiddo, "Combinatorial Optimization with Rational Objective Functions", **Mathematics of Operations Research**

4, pp 414-424, (1979).

17] A.V.Goldberg, and R.E.Tarjan "A New Approach to the Maximum Flow Problem", **Proc. 18th ACM Syomp. on the Theory of Comput., pp 136-146, (1986).**

[18] G.Finke, "A Primal Algorithm for class of Minimax Network Flow Problem"

[19] Arie Tamir "On the Solution of Discrete Bottleneck Problems", **Discrete Applied Mathematics 4, pp 299-302, (1982).**

[20] M.Minoux "Flots êquilibrês et flots avec sêcuritê", **E.D.F-Bulletein de la Direction des Etudeset Recherches - sêrie C,5-16.**

[21] S.Fujishige, A.Nakayama and W.Cui, "On Equivalence of the Maximum Balanced Flow Problem and the Weighted Minimax flow Problem", **Operations Research Letters 5, pp 207-209, (1986).**

[22] A.J.Goldman, "The Minimax Transportation Problem", **Transportation Science -2, pp 383-387, (1986).**

[23] R.K.Ahuja, "Algorithm for Minimax Transportation Problem", **Naval Research Logistics Quarterly 33, pp 725-739, (1986).**

[25] P.L.Hammer, "Time Minimizing Transportation Problems", **Naval Research Logistics Quarterly 16, pp 345-357, (1969).**

[26] V.Srinivasan and G.L.Thompson, "Algorithm for Minimizing Total Cost, Bottleneck time and Bottleneck Shipment in Transportation problems", **Naval Research Logistics Quarterly 23, pp 567-595, (1976).**

[27] R.A.Russel, D.D.Klingman, P.P.Navid, "An Efficient Primal Approach to Bottleneck Transportation Problems", **Naval Research Logistics Quarterly 30, pp 13-35, (1983).**

[28] A.M.Frieze, "Bottleneck Linear Programming", **Operations Research Quarterly 26, pp 345-357, (1979).**

[29] S.Martello, W.r.Pulleyblank, P.Toth, D.De.Werra, "Balanced Optimization Problems", **Operations Research Letters 3, pp 275-278, (1984).**

[30] R.K.Ahuja, "Balanced Linear Programming Problem", **Technical reports, I.I.T, Kanpur.**

[31] Katta G.Murty "Linear Programming ",**John Wiley and Sons, ISBN 0.471-09 725x T57.74.M87 519.7 2 83-7012, pp 288.**

[32] Roy E.Marsten, "The Design of XMP Linear Programming Library", **Transaction on Mathematical Software 7,**

No. 4, (1981).

D.A.Klingman, A.Napier, J.Stutz, "Netgen : A Programming for Generating Large Scale Capacited Assignment, Transportation and Minimum Cost Network Problem" **Management Science 20, pp 814-821, (1974).**

IMEP-1991-M-NAI-ALG